# Generalized Distance Functions

ERGUN AKLEMAN
Visualization Sciences
College of Architecture
Texas A&M University
College Station, Texas 77843-3137
email: ergun@viz.tamu.edu
phone: +(409) 845-6599
fax: +(409) 845-4491

JIANER CHEN
Department of Computer Science
College of Engineering
Texas A&M University
College Station, TX 77843-3112
email: chen@cs.tamu.edu
phone: +(409) 845-4259
fax: +(409) 847-8578

## Abstract

*In this paper, we obtain a generalized version of the well-known distance function family $L_p$ norm. We prove that the new functions satisfy distance function properties. By using these functions, convex symmetric shapes can be described as loci, the set of points which are in equal distance from a given point. We also show that these symmetric convex shapes can be easily parameterized. We also show these distance functions satisfy a Lipschitz type Condition. We provide a fast ray marching algorithm for rendering shapes described by these distance functions. These distance functions can be used as building blocks for some implicit modeling tools such as soft objects, constructive soft geometry, freps or ray-quadrics.*

## 1 Introduction and Significance

The concept of distance function has been developed to provide a formal description for measuring distance between two points in a vector space. Any function which satisfies the following logical conditions for distance can be used for measuring distance in a vector space.

**Definition 1.1** *Let $\mathcal{V}$ be a vector space and $\Re^*$ be the set of all non-negative numbers, a function $f : \mathcal{V} \longrightarrow \Re^*$ is a distance function, if it satisfies the following distance conditions:*
*(1) $f(v) = 0$ if and only if $v = 0$;*
*(2) $f(v) = f(-v)$;*
*(3) $f(v_1 + v_2) \leq f(v_1) + f(v_2)$ for any $v_1, v_2 \in \mathcal{V}$.*

There exist various distance functions defined over different vector spaces such as $L_p$ norm, Hausdorf-Besicowitch distance, and Hamming distance. $L_p$ norm is a particularly interesting distance function for modeling shapes. In 3-dimensional vector space, $L_p$ is given as

$$f(x, y, z) = \left(|x|^p + |y|^p + |z|^p\right)^{1/p}.$$

$L_p$ norm is based on the operator $\left(\sum_{i=1}^{r}()^p\right)^{1/p}$ which we call Minkowski operators. These operators satisfy well-known Minkowski's inequality. Interested readers can find a proof for this inequality in many classical mathematics textbooks (e.g., [15], page 55).

**Proposition 1.1 (Minkowski)** *If $a_i \geq 0$ and $b_i \geq 0$, for $i = 1, \ldots, r$, and $p \geq 1$, then*

$$\left(\sum_{i=1}^{r}(a_i + b_i)^p\right)^{1/p} \leq \left(\sum_{i=1}^{r} a_i^p\right)^{1/p} + \left(\sum_{i=1}^{r} b_i^p\right)^{1/p}$$

The Minkowski operators have been used in implicit shape modeling in quite a long time. Ricci [20] extended these operators by including negative $p$ values and obtained exact and approximate set operations which we call Ricci operators. Barr [3], independently, has developed superquadrics by using Minkowski operators. Hanson [11] also used Minkowski operators to generalize superquadrics to Hyperquadrics. Akleman [1] developed ray-linears, a function family that provide fast computation and closed under Ricci operators. Wyvill [24] has developed Constructive Soft Geometry also by using Ricci's operators. Even Rvachev's exact set operations [17] are related to Minkowski operators. Distance functions are also useful to generate implicit field functions from various types of shape information [13, 8, 9, 10]. In this paper, we obtain new distance functions also using Minkowski operations.

Distance functions are interesting for shape modeling for many reasons.

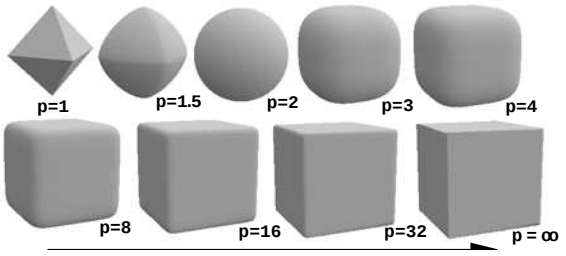


**Figure 1. Simple Locus with $L_p$ norm.**

- Locus of all the points which have equal distance to a given point gives various symmetric convex shapes when different distance functions are used. For instance, $L_p$ norm in 3-dimension gives shapes including octahedron, sphere and cube as shown in Figure 1.
- It can be shown that the loci defined above by using distance functions can be computed extremely fast and interactively be modified.
- Since these loci can be modified interactively, they can be used as building blocks in constructive (either solid or soft) geometry. In this way, the user can have an approximate idea how the overall shape will look like, before rendering the whole shape.
- The new distance functions we propose extend the number of building blocks considerably by including every symmetric convex polyhedra and their approximation.
- The distance functions satisfy a Lipschitz type condition. This condition provides a fast ray marching method for rendering shapes described by these distance functions.

## 2 Generalized Distance Functions

To provide generalized distance functions, we first show that the following simple functions, which are generalizations of $L_p$ norm, are distance functions.

### 2.1 Simple Case

Let $S = \{n_1, \ldots, n_r\}$ be a set of vectors, let $\mathcal{V} = span(S)$ and let $p \geq 1$ be a real number. We define a function $f_S$ from $\mathcal{V}$ to $\Re^*$ as follows.

$$f_S(v) = \left( \sum_{i=1}^{r} |n_i \cdot v|^p \right)^{1/p}$$

where $|n_i \cdot v|$ is the absolute value of the inner product of the vectors $n_i$ and $v$.

**Theorem 2.1** *The function $f_S$ defined above is a distance function.*

PROOF. By the definition, $v = 0$ implies $f_S(v) = 0$. On the other hand, suppose $v \neq 0$. Since $v \in \mathcal{V}$ and $\mathcal{V} = span(S)$, we can write it as $v = \sum_{i=1}^{r} a_i n_i$. If we multiply both sides with vector $v$, we get $v \cdot v = \sum_{i=1}^{r} a_i n_i \cdot v$. Since left side is always positive for any $v \neq 0$, at least one of the $n_i \cdot v$ should be non-zero. Then since at least one of the values $|n_i \cdot v|$ is nonzero, we have $f_S(v) > 0$. This proves (1).

Since $|n_i \cdot v| = |n_i \cdot (-v)|$ for any vectors $n_i$ and $v$, condition (2) holds.

Now we prove condition (3). We have

$$\begin{aligned}
f_S(v_1 + v_2) &= \left( \sum_{i=1}^{r} |n_i \cdot (v_1 + v_2)|^p \right)^{1/p} \\
&= \left( \sum_{i=1}^{r} |n_i \cdot v_1 + n_i \cdot v_2|^p \right)^{1/p} \\
&\leq \left( \sum_{i=1}^{r} (|n_i \cdot v_1| + |n_i \cdot v_2|)^p \right)^{1/p} \\
&\leq \left( \sum_{i=1}^{r} |n_i \cdot v_1|^p \right)^{1/p} + \left( \sum_{i=1}^{r} |n_i \cdot v_2|^p \right)^{1/p} \\
&= f_S(v_1) + f_S(v_2)
\end{aligned}$$

The first inequality follows from the fact $|a + b| \leq |a| + |b|$ for any real numbers $a$ and $b$, and the second inequality follows from Minkowski's inequality. □

The next section describes a generalized function and show that this function is also a distance function.

### 2.2 Generalized Case

We again let $S = \{n_1, \ldots, n_r\}$ be a set of $r$ vectors and $\mathcal{V} = span(S)$. Let $T$ be a tree of $r$ leaves. We say that the tree $T$ is a *labeled tree* (with respect to the set $S$) if each leaf of $T$ is labeled by a vector $n_i$ in $S$, and each internal node $\sigma$ of $T$ is labeled with a real number $p_\sigma \geq 1$. An example of labeled tree is shown in Figure 2.

A function $f_T$ from $\mathcal{V}$ to $\Re^*$ can be defined from the labeled tree $T$ as follows.

For a given vector $v$ in $\mathcal{V}$, each node $\sigma$ in $T$ is associated with a value $value(\sigma, v)$, which is recursively defined as follows. If the node $\sigma$ is a leaf labeled by a vector $n_i$, then $value(\sigma, v) = |n_i \cdot v|$; if $\sigma$ is an internal node labeled with a real number $p_\sigma \geq 1$ and the chidren of $\sigma$ are $\tau_1, \ldots, \tau_h$,

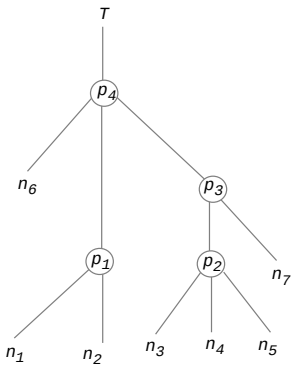


**Figure 2. An example of labeled tree.**

then

$$value(\sigma, v) = \left( \sum_{j=1}^{h} (value(\tau_j, v))^{p_\sigma} \right)^{1/p_\sigma}$$

The function $f_T(v)$ is defined to be equal to the value of the root of the tree $T$. Note that in particular, $f(v) = |n_i \cdot v|$ is a function in the space $span(n_i)$.

**Theorem 2.2** *Let $T$ be a labeled tree with respect to the vector set $S$. Then the function $f_T$ defined above is a distance function.*

PROOF. The proofs for distance conditions (1) and (2) from definition 1.1 are similar to that of Theorem 2.1, thus are omitted. We now prove condition (3).

The proof is by induction on the depth of the labeled tree $T$.

If the depth of the tree is 0, then the tree $T$ consists of a single node $\sigma$, which is labeled by a vector $n$. By the definition, $f_T(v) = value(\sigma, v) = |n \cdot v|$. Therefore, we have

$$\begin{aligned} f_T(v_1 + v_2) &= value(\sigma, v_1 + v_2) \\ &= |n \cdot (v_1 + v_2)| \\ &= |n \cdot v_1 + n \cdot v_2| \\ &\leq |n \cdot v_1| + |n \cdot v_2| \\ &= value(\sigma, v_1) + value(\sigma, v_2) \\ &= f_T(v_1) + f_T(v_2) \end{aligned}$$

Thus the theorem holds.

Now suppose that the labeled tree $T$ is of depth $d > 0$. Let $\sigma$ be the root of $T$, which is labeled $p_\sigma$, and suppose that $\tau_1, \ldots, \tau_h$ are the children of $\sigma$. Then

$$\begin{aligned} f_T(v_1 + v_2) &= value(\sigma, v_1 + v_2) \\ &= \left( \sum_{j=1}^{h} (value(\tau_j, v_1 + v_2))^{p_\sigma} \right)^{1/p_\sigma} \\ &= \left( \sum_{j=1}^{h} (f_j(v_1 + v_2))^{p_\sigma} \right)^{1/p_\sigma} \end{aligned}$$

Since the function $f_j$ is defined in terms of the labeled tree $T_j$ rooted at $\tau_j$ whose depth is less than $d$, by our inductive hypothesis, we have

$$f_j(v_1 + v_2) \leq f_j(v_1) + f_j(v_2)$$

This gives

$$f_T(v_1 + v_2) \leq \left( \sum_{j=1}^{h} (f_j(v_1) + f_j(v_2))^{p_\sigma} \right)^{1/p_\sigma}$$

Now applying Minkowski's inequality again, we get

$$\begin{aligned} f_T(v_1 + v_2) &\leq \left( \sum_{j=1}^{h} (f_j(v_1))^{p_\sigma} \right)^{1/p_\sigma} + \\ &\quad \left( \sum_{j=1}^{h} (f_j(v_2))^{p_\sigma} \right)^{1/p_\sigma} \\ &= \left( \sum_{j=1}^{h} (value(\tau_j, v_1))^{p_\sigma} \right)^{1/p_\sigma} + \\ &\quad \left( \sum_{j=1}^{h} (value(\tau_j, v_2))^{p_\sigma} \right)^{1/p_\sigma} \\ &= value(\sigma, v_1) + value(\sigma, v_2) \\ &= f_T(v_1) + f_T(v_2) \end{aligned}$$

This completes the proof. □

## 3 Shape Modeling with Distance Functions

The development of various distance functions is useful for defining shapes by using locus. The shape of locus of all the points which have equal distance to a given point depends directly on how we measure the distance. As shown in Figure 1, even for $L_p$ norm we can obtain various shapes with the same simple locus. Usage of our generalized distance functions considerably extends the class of shapes that can be generated by simple locus.

### 3.1 Simple Locus Defined with Generalized Distance Functions

Locus of all the points which have equal distance to a given point forms a symmetric convex shape because of the

distance conditions. Such symmetric convex shapes can be expressed by implicit representations

$$L(f_T, z, r) = \{v \mid f_T(v - z) - r = 0\}$$

where $f_T$ is the function we use for computing the distance, $z$ is the center and $r$ is the desired distance.

The shapes described by this simple locus can be easily understood by looking at the formula. Let $f_1(v - z)$ and $f_2(v - z)$ be two generalized distance functions which also include $|n_i \cdot (v - z)|$, from Ricci's constructive geometry [20], we know that the equation

$$max(f_1(v - z), f_2(v - z)) = r$$

gives the boundary of the intersection of the two shapes which are given by the inequalities $f_1(v-z) \leq r$ and $f_2(v-z) \leq r$. If we replace $max$ operation with Minkowski operations, we get an approximate version of the intersection, i.e., the sharp corners and edges resulted from exact intersection will be smoothed out. The smoothing operation can also be considered as a blending operation. The amount of blend can be controlled by the value of $p$. Lower $p$ values introduce more blend. The shape described by exact intersection by using $max$ operation can also be viewed as a control shape for the loci. For instance, in $L_p$ norm case, cube will be considered as a control shape for sphere.

By using this insight, it is easy to get an intuition for the locus. Since $|n_i \cdot (v - z)| = r$ is a symmetric stripe, the control shape will be a symmetric polyhedron described as an intersection of these symmetric stripes. According to the organization of labeled tree $T$ and $p$ values different smoothed version of the symmetric control polyhedron can be obtained. Figure 3 shows examples of symmetric stripes and symmetric control shapes in 2-dimensional space.

Figure 4 gives a list of simple loci in 3D based on various $f_T(v)$ type distance functions defined on 4,6,10,7 and 16 vectors respectively and different $p$ values. In the examples in Figure 4, we use simple distance function (the tree has only one internal node and $r$ leaves). Simple loci are obtained by using various subsets of following vectors:

$$\begin{aligned}
n_1 &= (1.000, 0.000, 0.000) \\
n_2 &= (0.000, 1.000, 0.000) \\
n_3 &= (0.000, 0.000, 1.000) \\
n_4 &= (0.577, 0.577, 0.577) \\
n_5 &= (-0.577, 0.577, 0.577) \\
n_6 &= (0.577, -0.577, 0.577) \\
n_7 &= (0.577, 0.577, -0.577) \\
n_8 &= (0.000, 0.357, 0.934) \\
n_9 &= (0.000, -0.357, 0.934) \\
n_{10} &= (0.934, 0.000, 0.357)
\end{aligned}$$

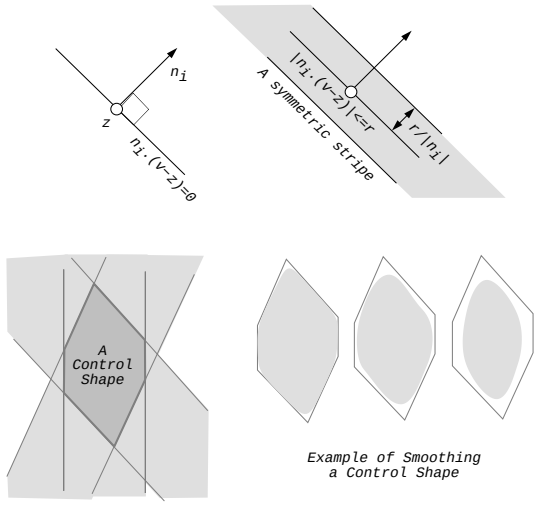


**Figure 3. Examples of symmetric stripes and control shapes.**

$$\begin{aligned}
n_{11} &= (-0.934, 0.000, 0.357) \\
n_{12} &= (0.357, 0.934, 0.000) \\
n_{13} &= (-0.357, 0.934, 0.000) \\
n_{14} &= (0.000, 0.851, 0.526) \\
n_{15} &= (0.000, -0.851, 0.526) \\
n_{16} &= (0.526, 0.000, 0.851) \\
n_{17} &= (-0.526, 0.000, 0.851) \\
n_{18} &= (0.851, 0.526, 0.000) \\
n_{19} &= (-0.851, 0.526, 0.000)
\end{aligned}$$

The simple loci of classical distance functions in earlier Figure 1 are obtained by using the vectors from $n_1$ to $n_3$.

Figure 5 shows the effect of choosing different trees over the shape of simple loci.

These simple loci are extremely easy to parameterize since generalized distance functions have an additional property that is called ray-linear which will be explained in the next subsection.

### 3.2 Ray-Linear Property of General Distance Functions

**Definition 3.1** *[1] Let $\mathcal{V}$ be a vector space. A function $f$ from $\mathcal{V}$ to $\Re^*$ is called ray-linear if it satisfies the ray-linear property $f(av) = af(v)$ for any vector $v$ in $\mathcal{V}$ and any positive real number $a$.*

Note that ray-linear function is not necessarily linear function. For example, $f(x, y) = (\sqrt{x} + \sqrt{y})^2$ is ray-linear

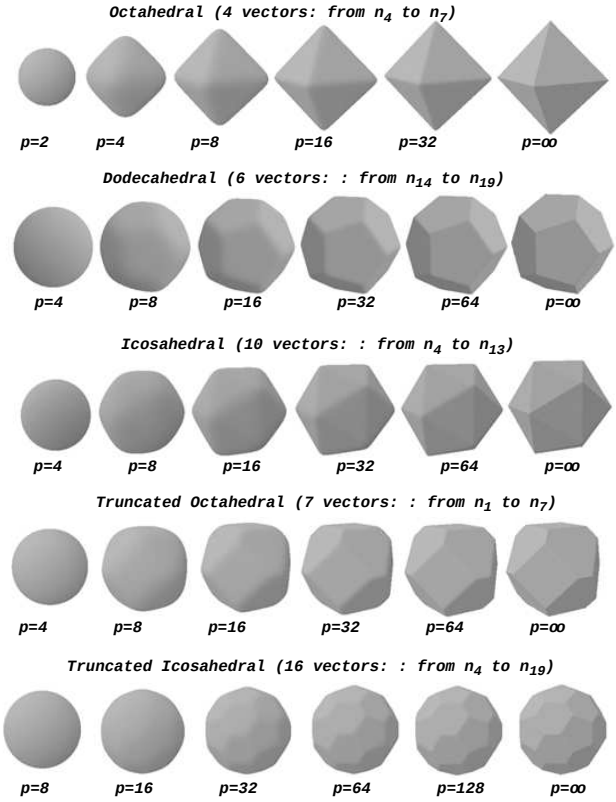


**Figure 4. Simple Loci for various sets of vectors and various $p$ values. Distance function is the one given in simple case, i.e., the tree has only one internal node and $r$ leaves.**

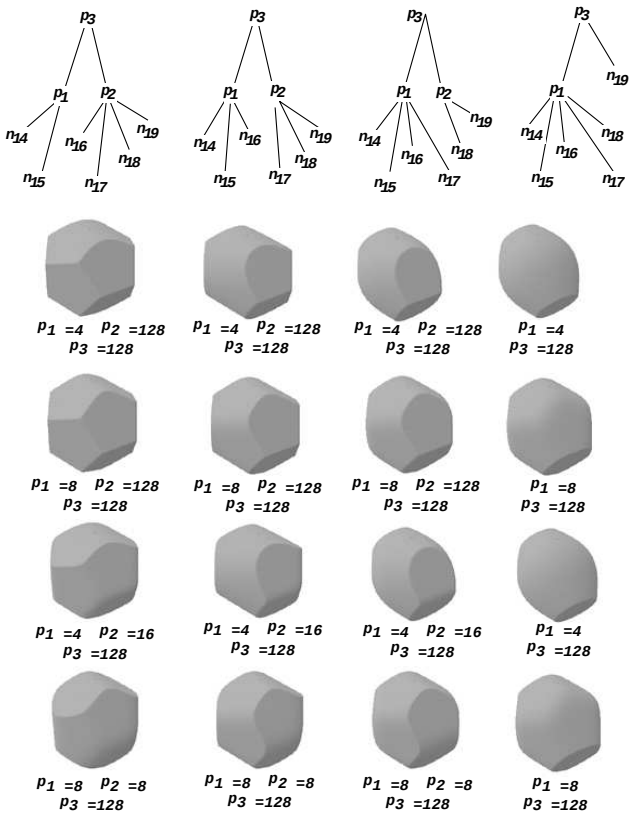


**Figure 5. Simple Loci with different trees.**

but not linear. Based on the following theorem [1, 2], it is easy to see that generalized distance functions satisfy the ray-linear property.

**Theorem 3.1** *Let $n_i$ be any vector in $\mathcal{V}$, then $f(v) = n_i \cdot v$ and $f(v) = |n_i \cdot v|$ are ray-linear. Let $f_1(v)$ and $f_2(v)$ be ray-linear functions, then $f_3(v) = \left(|f_1(v)|^p + |f_2(v)|^p\right)^{1/p}$ is also ray-linear.*

### 3.3 Parameterization of Simple Loci

Let a parametric representation of a unit hypersphere centered at $z$ be given as $v(s) = g(s) + z$ where $s$ is a vector in a given parameter space and $g(s)$ is a unit vector in $\mathcal{V}$. By using this parametric equation, we can write another parametric equation to represent a family of rays enamating from $z$ as $v(s,t) = g(s)t + z$. In the last equation, $t$ is a positive real number which actually gives the Euclidean distance between $v(s,t)$ and $z$. If we replace this equation of rays into equation of loci we get

$$f_T(g(s)t) - r = 0.$$

Because of the ray-linear property of $f_T$, the equation can be rewritten as

$$f_T(g(s))t - r = 0.$$

We can easily solve this equation for a given $s$ value as

$$t = \frac{r}{f_T(g(s))}.$$

If we plug in this $t$ value into the ray equation we get the desired parametric equation as

$$v(s) = \frac{r}{f_T(g(s))} g(s) + z.$$

Note that this equation is valid for any dimension and for any generalized distance function $f_T(v)$, as far as we have a parameterization of unit hypersphere. To create the images in Figures 1, 4 and 5, for parameterization of the shapes we used the following parameterization of a unit sphere:

$$g(s_1, s_2) = \begin{pmatrix} sin(2\pi s_1) sin(\pi s_2) \\ cos(2\pi s_1) sin(\pi s_2) \\ cos(\pi s_2) \end{pmatrix}$$

where $s_1, s_2 \in [0, 1]$.

### 3.4 Lipschitz Type Condition for Faster Ray Marching

Kalra and Barr [4] introduced Lipschitz condition into Computer Graphics for guaranteeing to find intersections with a ray and an implicit surface.

**Definition 3.2** *A positive real number $k$ is called a Lipschitz constant on a function $f : \mathcal{U} \longrightarrow \Re$ in a given region $\mathcal{U} \subseteq \mathcal{V}$, if given any two points $x, y \in \mathcal{U}$, the following condition holds:*

$$kd(y - x) \geq |f(y) - f(x)|$$

*where $d(.)$ is a vector norm. If the constant $k$ exists a Lipschitz condition is said to exist on the function $f$ in the region $\mathcal{U}$.*

Hart [12] also used Lipschitz condition to develop sphere tracing method and later applied Lipschitz condition to develop faster ray marching algorithm by letting longer steps in casting rays. The ray marching algorithm is originally introduced by Perlin & Hoffert [19] to ray cast implicitly represented volumes,

The distance functions in general (not only the ones we proposed in this paper) naturally satisfy a Lipschitz type condition. By using this condition, it is also possible to use long step sizes in ray marching.

**Theorem 3.2** *Every distance function $f$ satisfies the following Lipschitz type condition*

$$f(y - x) \geq |f(x) - f(y)| \quad \forall x, y \in \mathcal{V} \tag{1}$$

PROOF. Since $f$ is a distance function, it satisfies the inequality $f(y-x)+f(x) \geq f(y)$. If we take $f(x)$ to the right side of the inequality, we obtain the following inequality

$$f(y - x) \geq f(y) - f(x) \tag{2}$$

The distance function $f$ also satisfies the inequality $f(y) + f(x - y) \geq f(x)$. Because of the symmetry property of distance functions, we can rewrite this inequality as $f(y) + f(y - x) \geq f(x)$. If we take $f(y)$ to the right side of the inequality, we obtain

$$f(y - x) \geq -f(y) + f(x) = -(f(y) - f(x)) \tag{3}$$

By combining inequalities (2) and (3), we get Lipschitz type condition given in inequality (1). □

This theorem helps to develop a good sampling estimation for casting rays for rendering shapes defined by our distance functions. For instance, let a shape be given by simple locus $f(v - z) = r$ and a ray be given by a parametric function $v = v_1 t + v_0$. Let the intersection exist and be given by $t = t_{int}$. To optimize ray marching there is a need for the development of a method to obtain a series of $t_n$ values which quickly converges to $t_{int}$. In other words, the goal is to use the function value at $t = t_n$ to find a good estimate $t = t_{n+1}$ in such a way that $t_{n+1}$ can be as close as possible to $t_{int}$ without intersecting the surface. Although, we do not know the actual value of $t_{int}$, we do know the lower bound given by Lipschitz type condition. This lower bound can be obtained by using the inequality (1). First note that the left side of Lipschitz type condition is independent of $z$:

$$f(y - x) = f((y - z) - (x - z)) \geq |f(y - z) - f(x - z)|$$

If we plug in $y = v_1 t_{int} + v_0$ and $x = v_1 t_n + v_0$ in the above inequality

$$\begin{aligned} f((v_1 t_{int} + v_0) - (v_1 t_n + v_0)) &= \\ f(v_1(t_{int} - t_n)) &\geq \\ |f(v_1 t_n + v_0 - z) - f(v_1 t_{int} + v_0 - z)|. & \end{aligned}$$

Since $f(v_1 t_{int} + v_0 - z) = r$ we can rewrite the above inequality as

$$f(v_1(t_{int} - t_n)) \geq |f(v_1 t_n + v_0 - z) - r|.$$

Because of ray-linear property of our distance functions, we can rewrite the last inequality as

$$f(v_1)(t_{int} - t_n) \geq |f(v_1 t_n + v_0 - z) - r|.$$

By organizing this inequality, we obtain a lower bound for the value of $t_{int}$ as

$$t_{int} \geq t_n + \frac{|f(v_1 t_n + v_0 - z) - r|}{f(v_1)}$$

If we choose $t_{n+1}$ not larger than this lower bound, we can guarantee not to intersect with the surface,

$$t_{n+1} = t_n + \frac{|f(v_1 t_n + v_0 - z) - r|}{f(v_1)}$$

### 3.5 Generalized Loci and Lipschitz Type Condition

Locus of all the points which have the same sum of the distances to a set of given points gives a general loci. Such shapes can be expressed by implicit representation

$$\{ \ v \ | \ \sum_{i=1}^{N} a_i f_i(v - z_i) = r \ \}$$

where $f_i$ are any generalized distance functions, weights $a_i$ are positive real numbers, $z_1, z_2, \ldots z_N$ are points in $\mathcal{V}$ and

$r$ is the desired distance. The function $\sum_{i=1}^{N} a_i f_i(v - z_i)$ satisfies another Lipschitz type condition.

$$
\begin{aligned}
|\sum_{i=1}^{N} a_i f_i(x - z_i) - \sum_{i=1}^{N} a_i f_i(y - z_i)| &= \\
|\sum_{i=1}^{N} a_i (f_i(x - z_i) - f_i(y - z_i)| &\leq \\
\sum_{i=1}^{N} a_i |f_i(x - z_i) - f_i(y - z_i)| &\leq \\
\sum_{i=1}^{N} a_i f_i(x - y)
\end{aligned}
$$

Note that this inequality holds only for non-negative $a_i$ values and right side of the inequality is independent of $z_i$ values. Based on this result, it is easy to develop a ray marching method by inserting a ray equation $v(t) = v_0 + v_1 t$ ($v_1 \cdot v_1 = 1$) into the inequality. (However, these generalized loci cannot trivially be parameterized as we do for simple loci.) The resulting difference equation will be in the form of

$$t_{n+1} = t_n + \frac{|\sum_{i=1}^{N} a_i f_i(v_0 + v_1 t_n - z_i) - r|}{\sum_{i=1}^{N} a_i f_i(v_1)}$$

Figure 6 shows examples of generalized loci which are obtained by using two dodecahedral distance functions and rendered with the algorithm above.

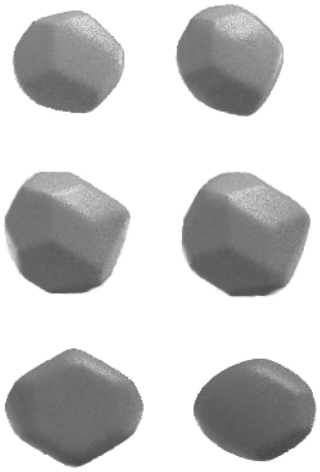

**Figure 6. Generalized Loci obtained by two dodecahedral distance functions.**

## 4 Distance Functions as Building Blocks

Since the simple loci defined by generalized distance functions can easily be computed by using their parameterized version, the generalized distance functions are extremely suitable to be used as building blocks for some implicit modeling tools such as soft objects, blobs and metaballs, constructive soft geometry, freps, or ray-quadrics [5, 14, 23, 22, 17, 2].

It is interesting to observe that when distance functions are used as building blocks of Blinn's exponentials [5] or Wyvill's soft objects [23], one of the concerns related to the use of distance functions can automatically be solved. This concern comes from the fact that the distance functions, because of their definitions, are not derivative continuous at the origin. However, when they are used in Blinn's exponentials [5] as in $e^{f(v)}$ or in Wyvill's soft object [22] functions $h()$'s as in $h(f(v))$, the resulting composite functions will automatically be derivative continuous at origin. In fact, this is an expected behaviour since both operations are initially suggested to be used over euclidean distance functions which are also not derivative continous at origin.

These distance functions fit especially the Constructive Soft Geometry [24] framework since Minkovski operations we use are a subset of Ricci operations which are used as Blending operators in *BlobTrees*.

## 5 Discussion and Conclusion

We obtain a generalized version of the well-known distance function family $L_p$ norm. We prove that the new functions satisfy distance function properties. By using this functions, convex symmetric shapes can be described as loci, the set of points which are in equal distance from a given point. We also show that these symmetric convex shapes can be easily parameterized. We also show these distance functions satisfy a Lipschitz type Condition. We provide a fast ray marching algorithm for rendering shapes described by these distance functions. These distance functions can also be used as building blocks.

We remark that the techniques presented for simple loci are also applicable for the generation of non-symmetric convex shapes. In order to create non-symmetric convex shapes from symmetric convex hyperquadrics, Hanson [11] observed that non-symmetric shapes can be obtained by intersecting the symmetric convex shapes with a lower dimensional hyperplane. For instance, the intersection of plane with a cube can create triangles, quadrilaterals and pentagons. The same idea can also be used to obtain non-symmetric convex 3D shapes by intersecting 4D simple loci with a 3D hyperplane. Moreover, The resulting shapes can still be computed by using the ray marching algorithm we presented.

## References


[1] E. Akleman, “Interactive Construction of Smoothly Blended Star Solids”, *Proceedings of Graphical Interface’96*, May, 1996.

[2] E. Akleman, “Ray-Quadrics”, *Proceedings of Implicit Surfaces’96*, pp. 89-98, Oct., 1996.

[3] A. H. Barr, “Superquadrics”, *IEEE Computer Graphics and Applications*, vol 1, no. 1, pp. 11-23, 1981.

[4] D. Kalra and A. H. Barr, “Guaranteed Ray Intersections with Implicit Surfaces”, *Computer Graphics*, vol 23, no. 3, pp. 297-306, 1989.

[5] J. I. Blinn, “A Generalization of Algebraic Surface Drawing”, *ACM Transaction on Graphics*, vol 1, no. 3, pp. 235-256, 1982.

[6] J. Bloomenthal, “Polygonization of Implicit Surfaces”, *Computer Aided Design*, No. 5, pp. 341-355, 1988.

[7] J. Bloomenthal and K. Ferguson, “Polygonization of Non-Manifold Implicit Surfaces”, *Computer Graphics*, vol 29, no. 4, pp. 309-316, 1995.

[8] C. Blanc and C. Schlick, “Extended Field Functions for Soft Objects”, *Proceedings of Implicit Surfaces’95*, pp. 21-32, Apr., 1995.

[9] C. Blanc and C. Schlick, “Implicit Sweep Objects”, *Computer Graphics Forum*, vol 15, no. 3, pp. 165-174, Aug., 1996.

[10] B. Crespin and C. Schlick, “Generating Implicit Field Functions from In/Out Images”, *Proceedings of Implicit Surfaces’98*, pp. 91-98, May, 1998.

[11] A. Hanson, “Hyperquadrics: Smoothly Deformable Shapes with Convex Polyhedral Bounds”, *Computer Vision, Graphics and Image Processing*, vol 44, no. 1, pp. 191-210, 1988.

[12] S. P. Worley and J. Hart, “Hyper Rendering of Hyper-Textured Surfaces”, *Proceedings of Implicit Surfaces’96*, pp. 99-104, Oct., 1996.

[13] Z. Kacic-Alesic and B. Wyvill, “Controlled Blending of Procedural Implicit Surfaces”, *Proceedings of Graphics Interface’91*, pp. 236-245, June, 1991.

[14] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I Shirakara and K. Omura, “Object Modeling by Distribution Functions”, *Electronic Communications*, vol 68, no. 4, pp. 718-725, 1985.

[15] D. S. MITRINOVIC, *Analytic Inequalities*, Springer-Verlag, New York, 1970.

[16] C. W. A. M. van Overveld and B. Wyvill “Shrinkwrap: An Adaptive Algorithm for Polygonization of an Implicit Surface”, *The University of Calgary, Department of Computer Science, Research Report No. 93/514/19*, March 1993.

[17] A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, “Function Representation in Geometric Modeling: Concepts, Implementations and Applications”, *Visual Computer*, no. 11, pp. 429-446, 1995.

[18] A. A. G. Requicha and H.B. Voelcker, “Solid Modeling: A Historical Summary and Contemporary Assessment”, *IEEE Computer Graphics and Applications*, vol 2, no. 2, pp. 9-24, March 1982.

[19] K. Perlin and E. M. Hoffert, “Hypertexture”, *Computer Graphics*, vol 23, no. 3, pp. 297-306, 1989.

[20] A. Ricci, “A Constructive Geometry for Computer Graphics”, *The Computer Journal*, vol 16, no. 2, pp. 157-160, May 1973.

[21] . B. T. Stander and J. Hart, “Guaranteeing Topology of Implicit Surface Polygonization for Interactive Modeling”, *Computer Graphics*, vol 31, no. 4, pp. 279-286, 1997.

[22] G. Wyvill and A. Trotman, “Ray Tracing Soft Objects”, *CG International ‘90: Computer Graphics Around the World*, Editors: T. S. Chua and T. L. Kunii, pp. 467-476, 1990, Springer Verlag.

[23] G. Wyvill, C. McPheeters, and B. Wyvill, “Data Structure for Soft Objects”, *The Visual Computer*, vol 2, no. 4, pp. 227-234, 1997.

[24] B. Wyvill, A. Guy, and E. Galin, “Extending The CSG Tree: Warping, Blending and Boolean Operations in an Implicit Surface Modeling System”, *Proceedigns of Implicit Surfaces’98*, pp. 113-121, 1998.