# A Bucket Grid Structure to Speed Up Table Lookup in Gauge-Based Photometric Stereo

Suppressed to avoid author identification

## Abstract

*In this paper, we show how to speed up the table lookup step in gauge-based multi-image photometric stereo. In that step, one must find a pixel of a* gauge object, *of known shape and color, whose appearance under $m$ different illumination fields is similar to that of a given scene pixel. This search reduces to finding the closest match to a given $m$-vector in a table with a thousand or more $m$-vectors. Our speed-up method explots the fact that the table is in fact a fairly flat two-dimensional manifold in $m$-dimensional space, so that the search can be efficiently solved with a two-dimensional bucket grid structure.*

## 1. Introduction

### 1.1. Variable-lighting photometric stereo

In *variable-lighting photometric stereo* (VLPS), the input data is a list of $m \geq 3$ monochromatic digital photos $S_1, .. S_m$ of some optically passive scene, all taken with different lighting conditions but with the same pose and viewpoint. As shown by R. J. Woodham in 1980 [8], by analyzing the $m$ pixel intensities $S_i[p]$ at any image point $p$, one can recover the unit vector $\hat{s}[p]$ that is perpendicular to surface element which is visible at $p$. This problem has attracted a lot of attention in recent years [3, 5, 7, 11, 2, 10].

To perform the above analysis, one must have enough information about the *bidirectional radiance distribution function* (BRDF) of the surface, and about the light field $\Phi_i$ in each image $S_i$. The BRDF of the scene's surface at point $p$ of the image is a function $\sigma[p](\hat{n}, \hat{u}, \hat{v})$ that gives the apparent brightness of the surface when oriented with normal $\hat{n}$, viewed from the direction $\hat{v}$, and illuminated with unidirectional light of unit intensity flowing in the direction $\hat{u}$. (Note that we include the geometric light spread factor $\max\{0, -\hat{u} \cdot \hat{n}\}$ in the BRDF itself.)

### 1.2. Gauge-based VLPS

In the *gauge-based* variant of VLPS, the BRDF information is indirectly given by $m$ images $G_1, .. G_m$ of a *light gauge*, a sample object of known shape and color; where each gauge image $G_i$ is taken under the same lighting conditions as the corresponding scene image $S_i$. (Depending on the application, it may be convenient to include the gauge object as part of the scene itself. In that case, each $G_i$ will be just a sub-image of $S_i$.)

In this paper, we assume that all images $S_1, .. S_m$ have been geometrically corrected, trimmed, and aligned, so that each point $p$ on their common domain $\mathcal{S}$ corresponds to the same point on the scene's visible surface. The same condition is assumed for the gauge images $G_1, .. G_m$, whose common domain will be denoted by $\mathcal{G}$. We also assume that all pixel values are directly proportional to light intensity.

In its basic form, gauge-based VLPS is viable only if all visible scene and gauge surfaces have the same finish everywhere, except for variations in intrinsic color. That is, the BRDF $\sigma[p]$ of the scene at each image point $p$, and the BRDF $\gamma[q]$ of the gauge at any point $q$, must be multiples of some fixed BRDF $\bar{\beta}$:

$$\begin{array}{rcl} \sigma[p](\hat{n}, \hat{u}, \hat{v}) & = & S^*[p]\, \bar{\beta}(\hat{n}, \hat{u}, \hat{v}) \\ \gamma[p](\hat{n}, \hat{u}, \hat{v}) & = & G^*[p]\, \bar{\beta}(\hat{n}, \hat{u}, \hat{v}) \end{array} \quad (1)$$

The constant factors $S^*[p]$ and $G^*[q]$ in these formulas are the *intrinsic lightness* or *albedo* of the scene and gauge surface, respectively, at those points. Observe that the gauge's albedo $G^*[q]$ and normal direction $\hat{g}[q]$ must be known for all $q \in \mathcal{G}$; typically, one uses a spherical gauge with uniform albedo, preferably white ($G^*[q] = 1$ everywhere).

Another necessary condition for gauge-based VLPS is that the BRDF $\bar{\beta}$ must be dominated by wide-angle scattering, with no mirror-like reflection or sharp glossy scattering. The standard example is the Lambertian BRDF

$$\bar{\beta}(\hat{n}, \hat{u}, \hat{v}) = \max\{0, -\hat{u} \cdot \hat{n}\} \quad (2)$$

However, almost any BRDF $\bar{\beta}$ will do, as long as it doesn't have impulse-like components (sharp peaks or ridges).

We will also assume that the the images are taken under nearly parallel projection and illuminated by distant light sources, so that the viewing direction $\hat{v}$ and the lighting conditions are the same at every point of $\mathcal{S}$ or $\mathcal{G}$.

Gauge-based VLPS can be extended to multichannel (e.g. RGB trichromatic) images, in which case one gets a single normal map $n$ but a different albedo map $S^*_\lambda[p]$ for each spectral band $\lambda$. These albedo maps then give the illumination-independent *intrinsic color* of the scene at each pixel.

## 1.3. Fundamental equations

The key idea of gauge-based VLPS is that a scene photo $S_i$ or a gauge photo $G_i$ can be factored into the product of two images: the *intrinsic albedo map*, $S^*$ or $G^*$, and a *lighting factor map* that depends only on the lighting conditions $\Phi_i$ and the local orientation of the surface. Specifically,

$$\begin{aligned} S_i[p] &= S^*[p]\, L_i(\hat{s}[p]) \\ G_i[q] &= G^*[q]\, L_i(\hat{g}[q]) \end{aligned} \tag{3}$$

Here, each $L_i$ is the *shading function* implied by the lighting field $\Phi_i$ and the BRDF $\bar{\beta}$. It maps each unit vector $\hat{n}$ to the apparent lightness of a white surface perpendicular to $\hat{n}$, and is given by the formula

$$L_i(\hat{n}) = \int_{\mathbb{S}^2} \Phi_i(\hat{u}) \bar{\beta}(\hat{n}, \hat{u}, \hat{v})\, \mathrm{d}\hat{u} \tag{4}$$

The factor $\Phi_i(\hat{u})$ is the intensity of the light flow that is incident on the surface from direction $\hat{u}$. The uniform lighting condition allows us to assume that $\Phi_i(\hat{u})$ does not depend on the position on the surface, but only on the light direction $\hat{u}$. Note that the value of $\Phi_i(\hat{u})$ is irrelevant for directions $\hat{u}$ that point outwards from the local surface, or for points that are not on the surface; so it is indeed plausible to have a single function $\Phi_i$ for the whole scene, independently of the local surface orientation. This lighting model allows attached shadows, and is adequate for scenes consisting of a single mostly convex object. The model cannot account for projected shadows, radiosity effects, or sources with uneven light distribution.

Note that, in this model, the intrinsic color maps $G^*$ and $S^*$ are distinct from each other, but are the same for all $i$; whereas the shading functions $L_i$ are different for each $i$ but are the same for $S_i$ and $G_i$.

If formulas (3) hold, then we can determine the normal $\hat{s}[p]$ at a point $p$ of the scene images by finding a point in the gauge images that reacts in the same way as $p$ to changes in lighting directions, except for the albedos $S^*[p]$ and $G^*[q]$. More precisely, we must find $q \in \mathcal{G}$ such that the $m$-vectors

$$\begin{aligned} \mathbf{S}[p] &= (S_1[p], S_2[p], \ldots, S_m[p]) \\ \mathbf{G}[q] &= (G_1[p], G_2[p], \ldots, G_m[p]) \end{aligned} \tag{5}$$

are multiples of each other. The vectors $\mathbf{S}[p]$ and $\mathbf{G}[q]$ are called the *observation vectors* (OVs) of points $p$ and $q$.

Having located the matching gauge point $q$, we can recover the normal vector $\hat{s}[p]$ and albedo $S^*[p]$ of the scene at $p$ by the formulas

$$\begin{aligned} \hat{s}[p] &= \hat{g}[q] \\ S^*[p] &= \frac{|\mathbf{S}(p)|}{|\mathbf{G}(q)|} G^*(q) \end{aligned} \tag{6}$$

This method will fail if there are two points $q', q''$ on the gauge images which have different normals ($\hat{g}[q'] \neq \hat{g}[q']$) but collinear OVs ($\mathbf{G}[q'] = \alpha\mathbf{G}[q'']$ for some scalar $\alpha$). To avoid this problem, the number of images $m$ must be at least 3, and the light fields $\Phi_1, .. \Phi_m$ must be sufficiently varied to break any such ambiguities. We will assume that this condition is satisfied in what follows.

## 1.4. The table-lookup step

The most time-consuming part of gauge-based VLPS is locating the point $q \in \mathcal{G}$ such that $\mathbf{S}[p]$ is a multiple of $\mathbf{G}[q]$. If neither vector is zero, this is equivalent to matching the *signatures* $\mathbf{s}[p]$ and $\mathbf{g}[q]$ defined by

$$\mathbf{s}[p] = \frac{\mathbf{S}[p]}{|\mathbf{S}[p]|} \qquad \mathbf{g}[q] = \frac{\mathbf{G}[q]}{|\mathbf{G}[q]|} \tag{7}$$

Here $|\cdot|$ is any norm of $\mathbb{R}^m$, e.g. the Euclidean norm

$$|\mathbf{X}| = \sqrt{\textstyle\sum_{i=1}^m X_i^2} \tag{8}$$

Note that the position $q$ is not meaningful by itself; it is only used to associate the signature $\mathbf{g}[q]$ to the normal $\hat{g}[q]$ and to the OV modulus $G^\bullet[q] = |\mathbf{G}[q]|$. Therefore, we can replace the gauge images by a *signature table*, an unordered set of triplets

$$\mathbb{T} = \{\, (\mathbf{g}[q], \hat{g}[q], G^\bullet[q]) : q \in \mathcal{G} \,\} \tag{9}$$

The computation of $\hat{s}[p]$ then becomes a *closest-match table look-up problem*, where we look for the element $(\mathbf{g}, \hat{n}, G^\bullet)$ of the table $\mathbb{T}$ that minimizes the distance $\operatorname{dist}(\mathbf{g}, \mathbf{s}[p])$.

The brute-force solution to this problem would be to scan the table $\mathbb{T}$, computing $\operatorname{dist}(\mathbf{g}, \mathbf{s}[p])$ for each signature $\mathbf{g}$ in it, while keeping track of the closest-matching entry. However, in order to provide a good coverage of all possible normal directions, the table $\mathbb{T}$ must have tens of thousands of entries. Since the lookup must be repeated for each pixel of the scene domain $\mathcal{S}$, it may take tens of minutes to process a single set of scene images with this method.

## 1.5. Previous work

A number of techniques have been proposed in the literature to speed up the table search step. In his pioneering work, Woodham [9, 10] used a regular $m$-dimensional grid spanning the hypercube $[0,1]^m$, with $2^b$ cells along each axis, for some bit count $b$. In the pre-processing phase, each gauge observation vector $\mathbf{G}[q]$ was quantized with $b$ bits per coordinate, yielding the $m$-tuple of indices of some grid cell where the associated normal vector $\hat{g}[q]$ was stored. (Woodham assumed uniform albedos $S^* = G^*$, so there was no reason to normalize the signatures.) In the lookup phase, each scene OV $\mathbf{S}[p]$ was mapped to a table cell in the same way, and the desired normal $\hat{s}[p]$ was recovered from the grid. One obvious disadvantage of this method is the size of the grid ($2^{mb}$, which is about 250,000 for $m = 3$ and $b = 6$).

Several researchers have used general $m$-dimensional nearest-point algorithms for this purpose. Hertzmann and Seitz [4] use *approximate nearest neighbour*(ANN) of Arya et al. [1]. Zhong and Little [11] use the *locally sensitive hashing* of Indyk and Motwani [6].

However, all the above methods have a common shortcoming: they consider the set of all gauge signatures $\mathbb{G} = \{\, \mathbf{g}[q] : q \in \mathcal{G} \,\}$ to be a generic cloud of points scattered in $m$-dimensional space, and therefore use general $m$-dimensional nearest-neighbor search algorithms, which are inherently expensive in space and/or time [6].

## 1.6. Shape of the signature table

The key observation for our improved method is that *the set $\mathbb{G}$ of all gauge signatures is essentially a two-dimensional subset of* $\mathbb{R}^m$. Therefore, we can reduce the problem to a two-dimensional nearest-point search, which can be solved very efficiently by a two-dimensional bucket grid scheme.

To understand the key observation above, note that, because of formulas (3) and (7), the normalized signatures $\mathbf{s}[p]$ and $\mathbf{g}[q]$ can be expressed as $\mathbf{l}(\hat{s}[p])$ and $\mathbf{l}(\hat{g}[q])$, respectively; where $\mathbf{l}$ is the *signature function*

$$\mathbf{l}(\hat{n}) = \frac{\mathbf{L}(\hat{n})}{|\mathbf{L}(hatn)|} \tag{10}$$

and $\mathbf{L}(\hat{n}) = (L_1(\hat{n}), .. L_m(\hat{n}))$. Note that the function $\mathbf{l}$, that maps surface normals to signatures, is defined only on the hemisphere $H$ of $\mathbb{S}^2$ consisting of the normal directions that deviate less than 90 degrees from the viewing direction $\hat{v}$. On the other hand, a good gauge object must provide a fairly dense and uniform sampling of $H$ (which is why spheres are normally used for that purpose). It follows that the set of gauge signatures must be a fairly dense and uniform cover of $K = \mathbf{l}(H)$, the range of the function $\mathbf{l}$.

Now, given our assumption that the gauge's BRDF $\bar{\beta}$ lacks the sharp spikes of mirror-like reflection, the shading factors $L_i(\hat{n})$ given by formula (4) are continuous functions of the surface normal $\hat{n}$. In fact, $L_i$ is typically fairly smooth, with just a few broad and hardly-distinguishable maxima. Observe, futhermore, that the gauge-based VLPS problem is solvable if and only if the function $\mathbf{l}(\hat{n})$ is invertible, i.e. for every point $\mathbf{v}$ of $\mathbb{S}^{m-1}$ there is at most one direction $\hat{n}$ such that $\mathbf{l}(\hat{n}) = \mathbf{v}$. If this condition holds, the range $K$ of $\mathbf{l}$ is an embedding of the hemisphere $H$ into $\mathbb{S}^{m-1}$. Finally, since the signatures are contained in the positive orthant of $\mathbb{R}^m$, the width of $K$, as seen from the origin of $\mathbb{R}^m$, is at most 90 degrees.

From these considerations, we expect (and experience confirms) that the range $K$ of $\mathbf{l}$ is a relatively flat and disk-like patch of a 2-dimensional manifold (surface) immersed in $\mathbb{S}^{m-1}$; and that the normalized gauge signatures $\mathbb{G}$ must be distributed over $K$ with fairly uniform density.

## 1.7. The 2D bucketing scheme

In our method, the signature table $\mathbb{T}$ is pre-processed as follows. We first compute the centroid $\mathbf{b}$ of the signature set $\mathbb{G}$, and find the unit vectors vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ that define the principal axes of the point cloud $\mathbb{G}$. These vectors are found by computing the $m \times m$ coordinate moment matrix $M$ of the displacements $\mathbf{g}_k - \mathbf{b}$, and taking the eigenvectors associated to its two largest eigenvalues. The point $\mathbf{b}$ and the vectors $\mathbf{u}, \mathbf{v}$ define a two-dimensional affine subspace $P$ of $\mathbf{R}^m$, the *signature projection plane*, which is roughly coplanar with the set $\mathbb{G}$. The orthogonal projection onto $P$ of a given normalized signature $\mathbf{g}$ will be denoted by $\downarrow \mathbf{g}$.

Next, we choose a regular grid of $N \times N$ square cells on the projection plane $P$. This grid is centered on the point $\mathbf{b}$, has its sides parallel to the vectors $\mathbf{u}$ and $\mathbf{v}$, and is barely large enough to contain the projection $\downarrow \mathbf{g}$ of any normalized signature $\mathbf{g}$ in $\mathbb{T}$. More precisely, the grid side is $2R$, where

$$R = \varepsilon + \max\{|(\mathbf{g}-\mathbf{b})\cdot\mathbf{u}|, |(\mathbf{g}-\mathbf{b})\cdot\mathbf{v}| : \mathbf{g} \in \mathbb{G}\} \tag{11}$$

for some small safety margin $\varepsilon$.

Having chosen the grid, we build, for each cell $C[i,j]$, a linked *bucket list* $T[i,j]$ of all table entries $(\mathbf{g}, \hat{n}, G^\bullet)$ whose signatures $\mathbf{g}$ project onto that cell. We also compute the corresponding *bucket mean* $\mu[i,j]$, defined as the barycenter of all signatures $\mathbf{g}$ in the list $T[i,j]$; and the *bucket radius* $\rho[i,j]$, defined as the the maximum Euclidean distance from $\mu[i,j]$ to any signature $\mathbf{g}$ in that list. See figure 1.

Note that the two-dimensional shape of $\mathbb{T}$ means that the entries in $T[i,j]$ are fairly close to each other, even if their mean distance from the plane $P$ is large compared to the cell size. This property remains true even when $m$ is greater than 3.

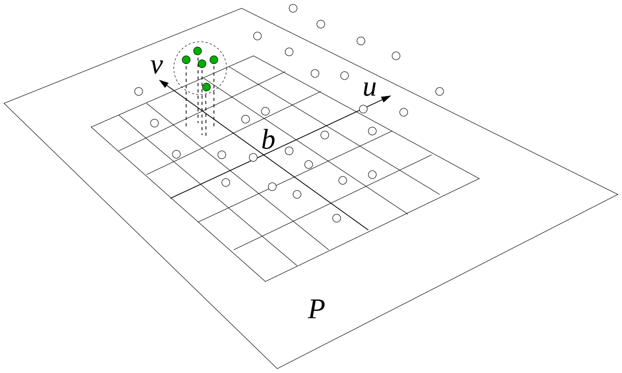


**Figure 1. The two-dimensional bucketing algorithm, for $m = 3$, showing some signatures in $\mathbb{G}$ (small circles), a bucket list $T[i,j]$ (small gray circles), and the enclosing sphere (dotted circle) defined by the bucket's centroid $\mu[i,j]$ and radius $\rho[i,j]$.**

Once the bucket grid has been constructed, the scene signatures $\mathbf{s}[p]$ are looked up with algorithm 1 below. Its steps are explained in sections 1.8 through 1.11.

**Procedure 1 (Table lookup)** *Given a signature* $\mathbf{s}$, *finds the entry tmin* $\in \mathbb{T}$ *whose signature* $\mathbf{t}$ *is most similar* $\mathbf{s}$.

```
1. $i \leftarrow \lfloor N((\mathbf{g} - \mathbf{b}) \cdot \mathbf{u} + R)/(2R) \rfloor$;
2. $j \leftarrow \lfloor N((\mathbf{g} - \mathbf{b}) \cdot \mathbf{v} + R)/(2R) \rfloor$;
3. dmin $\leftarrow +\infty$;
4. For each pair $(r,s)$ in $\Delta$, in order, do
   4.1. If dmin $\leq \delta \interleave (r,s) \interleave$, return tmin.
   4.2. $(i',j') \leftarrow (i,j) + (r,s)$;
   4.3. If $0 \leq i' < N$ and $0 \leq j' < N$, then
      4.3.1. For each $t = (\mathbf{g}, \hat{n}, G^\bullet)$ in $T[i',j']$, do
         4.3.1.1. If dmin $\leq \operatorname{dist}(\mathbf{s}, \mu[i',j']) - \rho[i',j']$,
                  finish step 4.3.1;
         4.3.1.2. set $d \leftarrow \operatorname{dist}(\mathbf{s}, \mathbf{g})$;
         4.3.1.3. If $d <$ dmin, set dmin $\leftarrow d$ and tmin $\leftarrow t$.
5. Return tmin.
```

## 1.8. Bucket grid searching

In order to locate the entry closest to a given normalized signature $\mathbf{s}$, we compute the indices $(i,j) = h(\mathbf{s})$ of the cell that contains its projection $\downarrow \mathbf{s}$. We then search for the entry $t$ whose signature $\mathbf{t}$ is closest to $\mathbf{s}$ in the list $T[i,j]$, and then, if necessary, in nearby buckets $T[i',j']$, in some appropriate order. Note that some buckets may be empty, and the best match to the query $\mathbf{s}$ may not be in bucket $T[i,j]$, even if that bucket is non-empty.

The bucket parameters $\mu[i,j]$ and $\rho[i,j]$ allow us to quickly skip over buckets that cannot possibly contain a better match to the query signature $\mathbf{s}$. More precisely, we can give up the search in a bucket $T[i',j']$ as soon as we can guarantee that the query signature $\mathbf{s}$ is closer to the best match $\mathbf{t}$ found so far than to any entry in that bucket. By the triangle inequality, this is ensured when

$$\operatorname{dist}(\mathbf{s}, \mathbf{t}) \leq \operatorname{dist}(\mathbf{s}, \mu[i',j']) - \rho[i',j'] \tag{12}$$

We will call condition (12) the *bucket truncation criterion*.

## 1.9. Precomputed search order

The bucket truncation criterion will often save us from looking at any entries of a bucket $T[i',j']$. However, if we were to examine the buckets in arbitrary order, we would have to check all $N^2$ buckets, and we would have to evaluate condition (12) for all of them.

To reduce this cost, we search the buckets $T[i',j']$ in a specific order, starting with the hashed bucket $T[i,j]$ and then moving gradually away from it. We are then able to determine when the best-match signature $\mathbf{t}$ has been found after scanning onlya fraction of the bucket array.

More precisely, consider two signatures $\mathbf{s}'$ and $\mathbf{s}''$ that project orthogonally to $P$ into cells $C[i',j']$ and $C[i'',j'']$, respectively. It is easy to see that

$$\operatorname{dist}(\mathbf{s}', \mathbf{s}'') \geq \operatorname{dist}(C[i',j'], C[i'',j'']) \tag{13}$$

In this formula, $\operatorname{dist}(C[i',j'], C[i'',j''])$ is the minimum distance between the two cells, seen as subsets of $P$. This distance is

$$\operatorname{dist}(C[i',j'], C[i'',j'']) = \delta \interleave (i' - i'', j' - j'') \interleave \tag{14}$$

where $\delta = 2R/N$ is the grid mesh size, and

$$\interleave (r,s) \interleave = \sqrt{(\max\{0, |r| - 1\})^2 + (\max\{0, |s| - 1\})^2} \tag{15}$$

Note that $\interleave (r,s) \interleave$ is a bit smaller than the Euclidean norm $|(r,s)| = \sqrt{r^2 + s^2}$. As part of the table preprocessing, we precompute an ordered list $\Delta$ of all pairs $(r,s)$ in $\{-N+1..N-1\} \times \{-N+1..N-1\}$, sorted by increasing value of $\interleave (r,s) \interleave$ (and breaking ties by $|(r,s)|$). For each query signature $\mathbf{s}$, we take each displacement $(r,s)$ from the list $\Delta$, in that order, and enumerate the bucket $T[i',j']$ where $(i',j') = (i,j) + (r,s)$ (provided that $i'$ and $j'$ lie in $\{0,..N-1\}$). See figure 2.

(a)

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | 5 | 4 | 4 | 4 | 5 | 8 |
| 5 | 4 | 1 | 1 | 1 | 4 | 5 |
| 4 | 1 | 0 | 0 | 0 | 1 | 4 |
| 4 | 1 | 0 | 0 | 0 | 1 | 4 |
| 4 | 1 | 0 | 0 | 0 | 1 | 4 |
| 5 | 4 | 1 | 1 | 1 | 4 | 5 |
| 8 | 5 | 4 | 4 | 4 | 5 | 8 |

(b)

| | | | | | | |
|---|---|---|---|---|---|---|
| 45 | 41 | 33 | 27 | 34 | 42 | 46 |
| 37 | 21 | 17 | 11 | 18 | 22 | 38 |
| 29 | 13 | 05 | 03 | 06 | 14 | 30 |
| 25 | 09 | 01 | 00 | 02 | 10 | 26 |
| 31 | 15 | 07 | 04 | 08 | 16 | 32 |
| 39 | 23 | 19 | 12 | 20 | 24 | 40 |
| 47 | 43 | 35 | 28 | 36 | 44 | 48 |

**Figure 2. (a) The squared cell distance function $\|\Delta\|^2$, and (b) the bucket scan order, for the $7 \times 7$ cells nearest to the starting cell (at center).**

### 1.10. Early termination

A bucket $[i', j']$ can be ignored if the cell distance bound (13) excludes the possibility that a better match can be found within it; that is, if

$$\operatorname{dist}(\mathbf{s}, \mathbf{t}) \leq \delta \|\Delta\| \qquad (16)$$

Note that condition (16) is weaker than condition (12); however, if condition (16) fails, we can stop the search and return $\mathbf{t}$, since that condition will fail for any subsequent $\Delta$.

### 1.11. Analysis

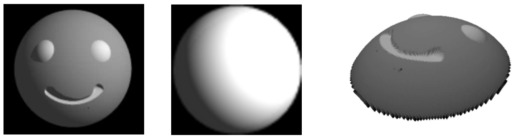
The average computation cost of algorithm 1 is roughly $Bb + Dd + O(1)$, where $b$ is the average number of buckets examined per lookup (step 4.1), $d$ is the average number of table entries tested (step 4.3.1.2), and $B, D$ are the costs associated to those two operations.

In the extreme case when $N = 1$, we will have $b = 1$ and $d = |\mathbb{T}|$ (which is equivalent to a linear search of $\mathbb{T}$). As $N$ increases, $d$ will usually decrease towards 1, because the test of step 4.1 will get satisfied before the procedure finds the second non-empty bucket. At the same time, $b$ will increase immediately to about 10, because $\|(r, s)\|$ is zero for the first nine pairs $(r, s)$ in the list $\Delta$. Thereafter, $b$ wil grow slowly in proportion to $N^2$, because the procedure will have to skip Increasingly more empty buckets before finding the first non-empty one.

This analysis indicates that there will be an optimal value of $N$ which minimizes the running time. The optimum depends on the cost ratio $B/D$. In our tests, we found that the total time was minimized when $N$ was about $2\sqrt{|\mathbb{T}|}$ (an average of 0.25 entries per bucket).

## 2. Experiments

To measure the actual performance of our bucketing scheme, we used synthetic images produced by standard ray-tracing. The scene consisted of a hemispherical smiley-like mask with convex eyes and concave mouth (both in low relief in order to avoid projected shadows), with various shades of matte gray finish. See figure 3(a). The gauge object was a sphere with white Lambertian finish; see figure 3(b).



(a) (b) (c)

**Figure 3. Scene (a) and gauge (b) used in the tests. Figure (c) is a 3D view of the height map obtained by integrating the scene slopes computed by the method described in this paper.**

In all tests, the lighting setup was a single point source located very far from the scene. The camera field-of-view was narrowed to provide near-parallel image projection. In the tests, we varied the camera-to-light angle $\theta$ (either 10 or 45 degrees), the number of input images $m$ (either 3, 5, or 30). The signature table size $|\mathbb{T}|$ was kept fixed at 10219. For each combination of parameters, we ran our bucket-based algorithm on the scene images with grid sizes $N = 202$ and $N = 143$, corresponding to average entry-to-bucket ratios $\kappa = |\mathbb{T}| / N^2$ of 25% and 50%. We also processed the same images with $N = 1$, which is essentially equivalent to the brute-force nearest-match algorithm.

Table 1 shows various average cost metrics for each table look-up operation: the number $b$ of buckets $T[i', j']$ that were examined, the number $d$ of table entries that were actually tested (i.e., the number of evaluations of $\operatorname{dist}(\mathbf{s}, \mathbf{g})$), and the look-up time $t$ in microsseconds. The tests were run on a standard PC with a 3GHz clock. The absolute time $t$ obviously depends on the implementation, so only the for the various configurations.

| $\theta$ | $m$ | $N$ | $\kappa$ | $t$ | $d$ | $b$ |
|---|---|---|---|---|---|---|
| 10° | 3 | 202 | 0.25 | 20.5 | 6.8 | 12.4 |
| 10° | 3 | 143 | 0.50 | 22.9 | 11.8 | 11.2 |
| 10° | 3 | 1 | — | 3987.5 | 10219.0 | 1.0 |
| 45° | 3 | 202 | 0.25 | 18.0 | 3.5 | 10.0 |
| 45° | 3 | 143 | 0.50 | 18.3 | 6.4 | 10.0 |
| 45° | 3 | 1 | — | 3985.3 | 10219.0 | 1.0 |
| 10° | 5 | 202 | 0.25 | 22.3 | 6.4 | 11.7 |
| 10° | 5 | 143 | 0.50 | 25.1 | 11.1 | 10.9 |
| 10° | 5 | 1 | — | 5620.1 | 10219.0 | 1.0 |
| 45° | 5 | 202 | 0.25 | 29.0 | 10.5 | 45.2 |
| 45° | 5 | 143 | 0.50 | 28.7 | 12.2 | 28.9 |
| 45° | 5 | 1 | — | 5606.3 | 10219.0 | 1.0 |
| 10° | 30 | 202 | 0.25 | 58.4 | 9.7 | 11.4 |
| 10° | 30 | 143 | 0.50 | 76.4 | 16.7 | 10.8 |
| 10° | 30 | 1 | — | 26637.9 | 10219.0 | 1.0 |
| 45° | 30 | 202 | 0.25 | 74.9 | 12.5 | 51.2 |
| 45° | 30 | 143 | 0.50 | 78.9 | 14.1 | 32.3 |
| 45° | 30 | 1 | — | 26605.5 | 10219.0 | 1.0 |

**Table 1. Average costs and operation counts of the table look-up procedure for various values of $\theta$, $m$, and $N$. The entries with $N = 1$ represent sequential table search (without any bucket-grid speed-up).**

Figures 4 and 5 show the sizes of the bucket lists $T[i, j]$ for two different values of $m$ (5, and 30) and two different light arrangements ($\theta = 10°$ and $\theta = 45°$). In both cases we had $N = 202$ and $|\mathbb{T}| = 10219$, corresponding to an average entry-to-bucket ratio $\kappa = 25\%$. Note that, in most cases, the signatures are distributed fairly evenly over a substantial fraction of the grid.

## 3. Conclusions and future work

Our bucket-grid scheme provides fast and accurate best-match table search, even for very large values of $m$. Our two-dimensional grid is more space-efficient than the general $m$-dimensional nearst-neighbor data structures used bu other authors. It is also considerably faster than those methods. Thanks to the optimal alignment of the grid, we obtain compact spherical enclosures for each bucket, which allow us to eliminate an entire bucket with a single distance comparison. Moreover, the 2D structure means that we need to scan only a few buckets (10 or so) around the hashed cell. Moerover, unlike previous grid schemes, our method is exact—it always yields the best matching entry in the table, and not merely a close approximation.

We have restricted the input to monochromatic images only to simplify the exposition; but our 2D bucket-grid method works equally well for color images. If each image has $c$ spectral bands (color channels), the *color observation vectors* $\mathbf{S}[p]$ and $\mathbf{G}[q]$ are the concatenation of $c$ monochromatic OVs with $m$ components each. As before, in order to recover the scene normal $\hat{s}[p]$ at a point $p$, we look for for a gauge point $q$ such that the color signatures $\mathbf{s}[p]$ and $\mathbf{g}[q]$ match; except that the color signatures are obtained from the color OVs by normalizing each monochromatic OV separately. The color signatures are then points of $(\mathbb{S}^m)^c$; but they are still a 2-dimensional manifold in that space, and therefore can be organized by a single 2-D bucket grid.

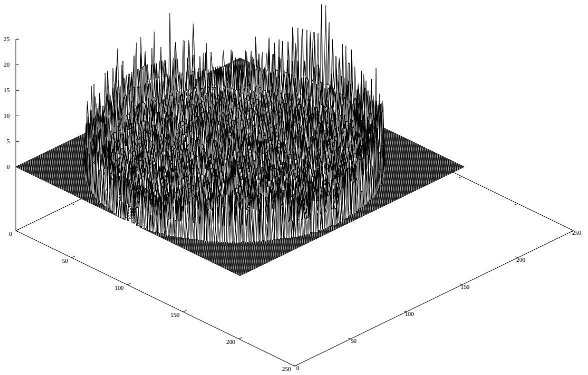


**Figure 4. Bucket list lengths for $m = 5$ and $\theta = 45°$. The longest bucket has 8 entries.**

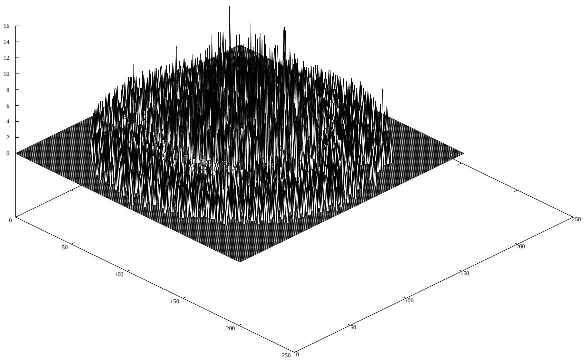


**Figure 5. Bucket list lengths for $m = 30$ and $\theta = 10°$. The longest bucket has 5 entries.**

## Acknowledgements

This project was partly supported by research grants from CNPq and CAPES.

## References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate neareast neighbor searching in fixed dimensions. *Journal of the Association of Computing Machinery*, 45(6):891–923, 1998.

[2] S. Barsky and M. Petrou. The 4-source photometric stereo technique for three-dimensional surfaces in presence of highlights and shadows. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(10):1239–1252, Oct. 2003.

[3] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general unknown lighting. *International Journal of Computer Vision*, 72(3):239–257, 2007.

[4] A. Hertzmann and S. M. Seitz. Shape and materials by example: A photometric stereo approach. In *Proceedings IEEE CVPR 2003*, volume 1, pages 533–540, June 2003.

[5] A. Hertzmann and S. M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying BRDFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, Aug. 2005.

[6] P. Indyk and R. Motwani. Approximate nearest neighbour: Towards removing the curse of dimensinality. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 604–613, 1998.

[7] L. Shen, T. Machida, and H. Takemura. Efficient photometric stereo for three-dimensional surfaces with unknown BRDF. In *Proceedings of the 5th International Conference on 3-D Digital Imaginga nd Modeling (3DIM'05)*, pages ???–???, 2005.

[8] R. J. Woodham. Photometric method for determining suface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[9] R. J. Woodham. Determining surface curvature with photometric stereo. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, volume 1, pages 36–42, May 1989.

[10] R. J. Woodham. Gradient and curvature from the photometric stereo method, including local confidence estimation. *Journal of the Optical Society of America, Series A*, 11(11):3050–3068, 1994.

[11] L. Zhong and J. J. Little. Photometric stereo via locality sensitive high-dimension hashing. In *Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV'05)*, pages ???–???+7, 2005.