MVA2000 IAPR Workshop on Machine Vision Applications, Nov. 28–30, 2000, The University of Tokyo, Japan

13—6

Fast Lighting/Rendering Solution for Matching a 2D Image to a Database of 3D Models: "Lightsphere"

A. Peter Blicher *
NEC Research Institute

Sébastien Roy †
Département d'Informatique et recherche opérationnelle
Université de Montréal

Abstract

We describe a method for object recognition with 2D image queries to be identified from among a set of 3D models with known pose. The main target application is face recognition. The 3D models consist of both shape and color texture information, and the 2D queries are color camera images. The kernel of the method consists of a lookup table that associates 3D surface normals with expected image brightness, modulo albedo, for a given query. This lookup table is fast to compute, and is used to render images from the models for a sum of square difference error measure. Using a data set of 42 face models and 1764 (high quality) query images under 7 poses and 6 lighting conditions, we achieve average recognition accuracy of about 80%, with more than 90% in several pose/lighting conditions. The method is extremely fast compared to those that involve finding eigenvectors or solving constrained equation systems.

1 Introduction

We are interested in searching a potentially very large database of 3D solid models including texture to find a match to a 2D photographic query, under conditions of arbitrary illumination and pose. We have developed a technique to accomplish this using data captured by rangefinder hardware developed by our collaborators at the NEC C&C Media Research Lab in Miyazaki-dai, Japan. This hardware[3] (an early version of the Fiore model) captures accurate shape information in registration with texture data, producing a 640×640 texture and range mesh, with 24 bit color and less than 1mm range error.

Our technique, which we call "lightsphere", is based on a recognition paradigm which uses computer graphics techniques to render from the model database and then make a comparison with the query image. In this paradigm, pose is first determined by some means, then based on that pose and knowledge of the query, an estimate is made of the appearance of each model under the same lighting conditions as the query. The model whose rendered appearance is estimated to most closely match the query is deemed the most likely identification.

Even when the correct pose of a 3D model is known, the illumination can create large differences between the reprojected 3D model and the query image, so that properly accounting for lighting variations is a major challenge.

In a very large database, the computational cost of comparing many models to a query is critical. The lightsphere method has been designed to be very fast and is not very restrictive in the assumptions it makes about the albedo, the surface properties, and the lighting conditions.

Lightsphere avoids solving for light sources, as one might do e.g. following the methods of Georghiades, Kriegman, and Belhumeur[1]. We believe this offers a substantial speed improvement. (In our preliminary testing, comparing with naive implementations of light solving using nonnegative least squares methods, we observe a factor 10–100 speedup.) We envision multiple matching methods operating in a sifting process, so that our faster method would filter out the great majority of false matches, sending the most promising candidates to more costly processes for more precise analysis.

The lightsphere method assumes that pose has already been solved at a previous stage. In our present system, we use pose based on a solution from a small number of hand-extracted feature points. Given a query, pose must be solved for each possible model in the database; however this is quite fast.

The kernel of the lightsphere algorithm is based on the fact that given a 2D query, a candidate 3D model, and a corresponding pose, we can project the 3D model in registration with the 2D query. Of course, we may be matching (or aligning) the wrong 3D model, in which case we would expect the reg-

*Address: 4 Independence Way, Princeton, NJ 08540-6634 USA. E-mail: blicher@research.nj.nec.com

†Address: CP 6128 Succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada. E-mail: roys@iro.umontreal.ca

istration to be poor. If one knew the lighting for the query, then one could render the 3D model in registration with the 2D query, and compute an error measuring the similarity between the rendered image and the query image.

A possible approach is that one could use the normals and the texture from the 3D model, in conjunction with the assumption of Lambertian reflectance, to compute a light source distribution that best accounts for the query, and we have also tried that approach; however, the lightsphere method is able to avoid the costly step of actually solving for the light sources, as follows.

2 Lighting Model and Notation

We assume that all light sources are at infinity (i.e., isotropic lighting), so that the lighting does not vary from point to point on the illuminated surface (except for the effect of self-shadowing by attached shadows). We make no assumption about the type of surface of the object (e.g., we do not assume that it is Lambertian). Rather, we assume only that the reflected light depends on the surface normal, the incoming light, and linearly on some albedo that may vary from point to point on the surface.

Let:

A be the albedo (intrinsic reflectance) function,

N be the Gauss map of the surface, i.e., the function that maps each point to its normal.

L be a function on directions in 3-space that represents the light intensity coming from each direction.

I be the observed image intensity on the surface.

I.e., if we call the surface S, then

$A : S \to \mathbf{R}^+$, where $\mathbf{R}^+$ is the nonnegative real numbers;

$N : S \to G$, where G is the Gaussian sphere;

$L : G \to \mathbf{R}^+$; and

$I : S \to \mathbf{R}^+$. [1]

In this notation, the Lambertian model of surface reflectance with attached shadows can be written as

$$I = A \int_{g \in G} L(g) \rho \circ (g \bullet N) \qquad \text{(eqn L)}$$

[1] Notice that we are using notation where these entities are functions; for example, the albedo at the point $p \in S$ is written as $A(p)$ in this notation, and the light intensity from the direction $g \in G$ is written as $L(g)$.

where $\bullet$ is the usual vector dot product; ρ is a rectifier function which clamps negative values to zero, i.e. $\rho(x) \equiv max(0, x)$; and the function I is defined by

$$I(p) = A(p) \int_{g \in G} L(g) \rho(g \bullet N(p)), \quad p \in S \qquad \text{(eqn Lp)}$$

3 The Lightsphere Method

Note that eqn. Lp is of the form

$$I(p) = A(p)\, B_L(N(p)), \qquad \text{(eqn B)}$$

with $B_L : G \to \mathbf{R}^+$. B can be thought of as a "brightness" function that depends only on L and $N(p)$, which captures the interaction of the lighting distribution with the normal. (For a Lambertian surface and a point source, this is just the dot product.) We assume only that the reflected intensity is governed by a relation of the form of eqn. B; we do *not* require that the surface be Lambertian. Notice, however, that this does presume that the reflected intensity does not depend on camera position.

In the render-compare recognition paradigm we are using, we are given A_{model}, N_{model}, and I_{query}, and we must compute an I_{rendered} from the model to compare to the query. In addition, we know the pose, and therefore we can register the model with the query (even for a wrong model).

Although L is unknown, we can compute the following quantity from our data, by dividing eqn. B to give

$$B_L(N_{\text{model}}(p)) = \frac{I_{\text{query}}(p)}{A_{\text{model}}(p)}, \qquad \text{(eqn LS)}$$

where we have identified points in the query with those in the model by using the pose information for registration.

Then we can render from a model, simply by multiplying B by the albedo A_{model}, yielding

$$I_{\text{rendered}}(p) = A_{\text{model}}(p) \cdot B_L(N_{\text{model}}(p)) \quad \text{(eqn A)}$$

If we were to do this point by point on the surface, this is a triviality, and we would simply get back the I_{query} that we started with. However, the true B_L depends only on the normal, and only implicitly on the location (through the map N_{model}). This imposes a constraint on the values that B_L can take, which we can exploit: different points with the same normal should have the same value of B_L. We therefore expect this constraint to be violated much more severely when matching the wrong model than when matching the correct model.

To get all the points with the same normal, consider the inverse of the Gauss map, N^{-1}. Then for

each $g \in G$, $N^{-1}(g)$ is the set of surface points with the normal g. The constraint says that under ideal conditions, for the model which matches the query, we would compute a B_L that is constant on each $N^{-1}(g)$. Of course, in the presence of noise and other errors this will not be exactly true, so instead we consider the average value of B_L computed on each $N^{-1}(g)$; call this $\bar{B}_L$. I.e.,

$$\bar{B}_L(g) = \frac{1}{|N^{-1}(g)|} \sum_{p \in N^{-1}(g)} \frac{I_{\text{query}}(p)}{A_{\text{model}}(p)}$$

Now we can render using

$$I_{\text{rendered}}(p) = A_{\text{model}}(p) \cdot \bar{B}_L(N_{\text{model}}(p)) \quad \text{(eqn } \bar{A}\text{)}$$

By now using $\bar{B}_L$, we get not a triviality, but a rendered image that should be faithful for the matching model (up to the limits of other errors), and poor for non-matching models.

In order to compute $\bar{B}_L$, we tesselate the Gaussian sphere of the model into bins, and compute $\bar{B}_L$ for each bin, simply by iterating across the query raster, looking up the corresponding normal and albedo from the model data, and accumulating a $\bar{B}_L$ for each bin.[2] (This Gaussian sphere bin data structure is the "lightsphere.")

Rendering the model in the lighting of the query then simply requires a lookup of $\bar{B}_L$ to insert into eqn. $\bar{A}$. However, in order to compensate for the quantization noise, we perform a bilinear interpolation on the $\bar{B}_L$ values depending on where the model normal vector to be rendered falls within a bin. This interpolated $\bar{B}_L$ is what is actually used for rendering.

When we render using the average of a bin, the pixel intensity error is a measure of how consistent the query and model are under the lighting that created the query. This can therefore be thought of as a cheap approximation to computing the mutual information between the query and the projected model, a technique elaborated in [2].

3.1 A note about albedo estimation

We have found that it is difficult to get accurate estimates of true albedo. Fortunately, the linearity of eqn. B provides us with a certain amount of robustness against inaccurate albedo estimates. We have found it useful to estimate albedo simply by applying a diffuse lighting. In the absence of cast shadows, the albedo can be measured as the image intensity resulting from perfectly diffuse lighting. However, in practice the lighting is not perfectly diffuse. Nevertheless, to the extent that the observed intensity obeys the relation of eqn. B, this is not a problem. Call the true albedo A^*, and the true "brightness" function B_L^*. Then the quantity we are using as the albedo, A, is really given by $A = A^* B^*_{L_A}$, where $B^*_{L_A}$ is the true "brightness" function for the lighting conditions L_A under which A was measured. Thus, $I = A \cdot (B_L^*/B^*_{L_A})$, which is again of the form of eqn. B, albeit with a B_L that is not the "real" brightness function, but due to linearity, this does not affect the result of estimating B_L and using it to render with A as albedo.

[2] Although we currently use a simple-minded checkerboard tesselation of the x-y plane, one could use vector quantization to optimize the bins. Although vector quantization is expensive, this can be computed offline for each model.

4 Results

We have tested this technique using a database of 42 3D models of human faces and 1764 queries, under all combinations of 7 poses and 6 lighting conditions, some of which are quite extreme. We achieve the recognition accuracies shown in Figure 1 with a preliminary (untuned) version of the algorithm.[3] See Figure 2 for the distribution of correct answer ranks, and Figures 3 and 4 for image examples of the algorithm.

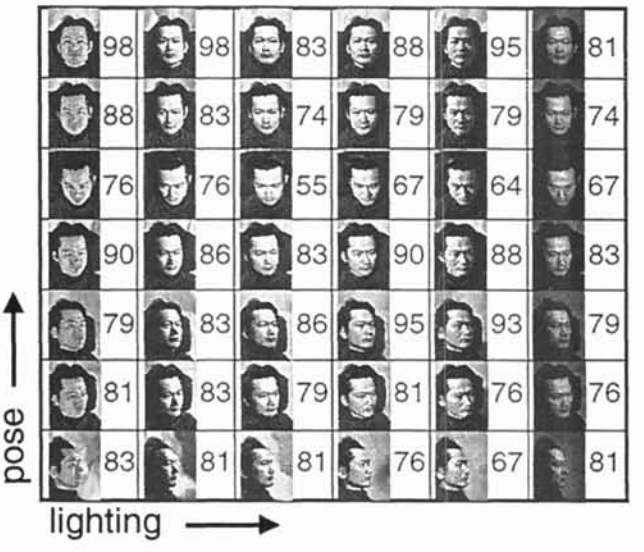

Figure 1: Percent recognition accuracy as a function of lighting and pose of the query. Images shown are reduced monochrome versions of full color query images for one subject. The query set consists of pictures of the same 42 individuals as used in the 3D model set.

Acknowledgments

We are deeply indebted to Johji Tajima, Shizuo Sakamoto, and Rui Ishiyama of the NEC C&C Me-

[3] We have found that a significant source of error is the pose solutions that are used; these numbers can be expected to improve with readily obtained better pose solutions.

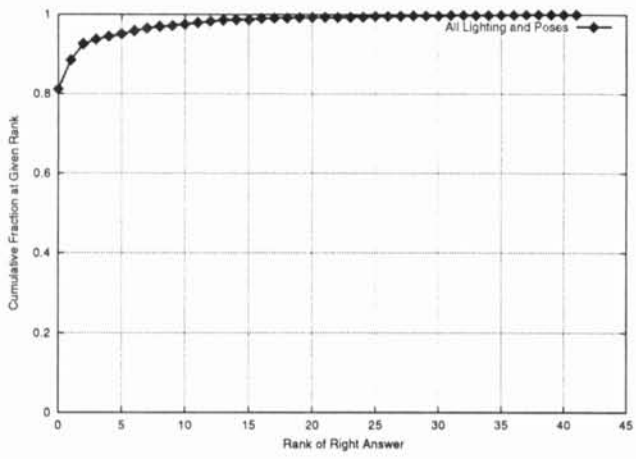

Figure 2: Cumulative rank curve taken over all 42 combinations of pose and lighting conditions. The ordinate represents the fraction of the time that the correct answer was ranked at or above the value on the abscissa. (Rank numbering starts at 0, not at 1.)

Figure 3: Example of operation of Lightsphere for correct match, i.e. when the model being examined is that of the individual in the query. From left to right: **(a)** query image; **(b)** 3D model texture projected for previously found pose for this query/model pair; **(c)** one component of the surface normal in registration with the model in (b); **(d)** lightsphere bin contents for R,G,B channels, and bin counts; **(e)** synthetic image of model in (b) rendered using computed lightsphere; **(f)** error image between (a) and (e), where lighter represents higher error.

dia Research Laboratory for generating and providing the data used in these experiments, as well as for many informative discussions that provided the

Figure 4: Example of operation of Lightsphere for wrong match — the model is of a different individual than the query. The query is the same as that in Figure 3, but a different model is being used. See Figure 3 for an explanation of subparts. Note the significantly larger error than in Figure 3 (which may reproduce poorly in printing). The synthetic image has been rendered based on normals from the (wrong) model, and lightsphere values from the intensity values of the query.

insight and inspiration for the ideas presented here. David Jacobs of the NEC Research Institute was instrumental in our approach to this problem, and in helping us see our technique in perspective (figuratively). C. W. Gear, David Waltz, and Mitsuhito Sakaguchi of the NEC Research Institute encouraged and made possible the collaboration with our Japanese colleagues whose fruits are reported here. Penio Penev of the Rockefeller University suggested the use of vector quantization for binning. We thank Shizuo Sakamoto and Rui Ishiyama for the kind permission to use their likenesses in the figures.

References

[1] Georghiades, Athinodoros S. , D. Kriegman, and P.N. Belhumeur, "Illumination Cones for Recognition under Variable Lighting: Faces," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Santa Barbara, 1998, 52–59.

[2] Viola, Paul A., Wells, III, William M., "Alignment by Maximization of Mutual Information," International Journal of Computer Vision (IJCV) (24), No. 2, September 1997, 137–154.

[3] `http://www.nec-eng.co.jp/cm/finder`