IEICE TRANS. INF. & SYST., VOL.E84–D, NO.12 DECEMBER 2001

LETTER *Special Issue on Machine Vision Applications*

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

Albert Peter BLICHER† *and* **Sébastien ROY**††, *Nonmembers*

SUMMARY We describe a method for object recognition with 2D image queries to be identified from among a set of 3D models. The pose is known from a previous step. 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 83%, with more than 90% in several pose/lighting conditions, using semi-automatically computed poses. The method is extremely fast compared to those that involve finding eigenvectors or solving constrained equation systems.

key words: *face recognition, biometrics, computer vision, model-based vision*

1. Introduction

We are interested in searching a potentially very large database of 3D solid models, which include texture information, 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 Kawasaki, Japan. This hardware [6], [13], (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 0.5 mm range error.

Our approach 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 compensating for lighting variations is a major challenge for recognition. Major contributions to understanding the interplay of lighting, 2D appearance, and 3D shape have been recently made by Belhumeur, Kriegman, and their coworkers [2]–[5] as well as others such as Vetter [11].

In a very large database, the computational cost of comparing many models to a query is critical. Our technique, which we call "lightsphere," 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 [5]. 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. Lightsphere is linear in query size and database size.)

Another approach to avoid solving for light sources is to use Principal Components Analysis (PCA) to find a subspace in which most of the energy of images under variable lighting lies [6], based on earlier work going back to [7], [9], and [10], though it should be noted that these earlier works were not concerned with lighting variation per se.

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 — 12 or fewer, depending on their visibility in the particular pose. Given a query, pose must be solved for each possible model in the database; however this is quite fast. In the applications we have in mind, a single query to be identified is sufficiently important that 2 or 3 minutes of human operator time is acceptable to click on these feature points. This produces pose solutions good enough to achieve fairly good results. Preliminary work shows that other methods of improving pose solutions greatly

Manuscript received February 20, 2001.
Manuscript revised June 15, 2001.

†The author is with the NEC Research Institute, 4 Independence Way, Princeton, NJ 08540-6634 USA.
††The author is with the Université de Montréal, Département d'Informatique et recherche opérationnelle, CP 6128 Succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada, and NEC Research Institute.

*This paper is an expansion of a paper that was presented at the IAPR Workshop on Machine Vision Applications (MVA2000), November 2000, Tokyo [1].

improve the accuracy lightsphere attains. That work is not reported here, however.

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 registration 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.

An important consequence of using a 3D model as a candidate to match the 2D query is that segmentation of the query is not needed, because each 3D model has already been segmented from the background in the 3D model data capture process. In our experience, it is easy to do this segmentation of the 3D model with minimal manual intervention, since the 3D data capture takes place under controlled conditions, and additionally the 3D information simplifies segmentation. When the 3D model and the 2D query are compared, they are compared under a known pose, and the segmentation of the 3D model is used as a mask to segment the 2D query. In the case where we are comparing the *wrong* model to the query, this segmentation of the 2D query will usually be *incorrect*, but this is a *desirable* property, because incorrect segmentation for the wrong model leads to *larger* error, and we want larger error when the model is incorrect.

In this context, one possible approach that can be imagined is to 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. This is similar to [5]. We have also tested 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
$I : S \to \mathbf{R}^+$.

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

$$I = A \int_{g \in G} L(g) \rho \circ (g \bullet N)\, d\mu \tag{1}$$

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

$$I(p) = A(p) \int_{g \in G} L(g) \rho(g \bullet N(p))\, d\mu, \quad p \in S \tag{2}$$

3. The Lightsphere Method

Note that Eq. (2) is of the form

$$I(p) = A(p)\, B_L(N(p)), \tag{3}$$

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. (In the special case of a Lambertian surface and a point light source, B_L is just the dot product with the light source vector, clipped by ρ to prevent negative brightness.) We assume only that the reflected intensity is governed by a relation of the form of Eq. (3); we do *not* require that the surface be Lambertian. Notice, however, that this does presume that the observed intensity at a point 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 — this is the best registration based on our pose estimate).

Although L is unknown, we can compute the following quantity from our data, by dividing Eq. (3) to give

$$B_L(N_{\text{model}}(p)) = \frac{I_{\text{query}}(p)}{A_{\text{model}}(p)}, \tag{4}$$

[†]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 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

$$I_{\text{rendered}}(p) = A_{\text{model}}(p) \cdot B_L(N_{\text{model}}(p)) \qquad (5)$$

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.

Formally, we can specify all the points with the same normal by considering 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.,

$$\bar{B}_L(g) = \frac{1}{|N^{-1}(g)|} \sum_{p \in N^{-1}(g)} \frac{I_{\text{query}}(p)}{A_{\text{model}}(p)} \qquad (6)$$

Now we can render using

$$I_{\text{rendered}}(p) = A_{\text{model}}(p) \cdot \bar{B}_L(N_{\text{model}}(p)) \qquad (7)$$

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[†]. (This Gaussian sphere bin data structure is the "lightsphere.")

The tesselation groups many similar normals together; therefore to be precise we must modify Eq. (6) slightly, as follows.

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

where Γ is now a region of the Gaussian sphere from our tesselation.

Rendering the model in the lighting of the query then simply requires a lookup of $\bar{B}_L$ to insert into Eq. (7). 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 [12][††].

Note that the lightsphere data structure of bins on the Gaussian sphere is a lookup table for a "brightness" coefficient that can be used in conjunction with the known albedo and normal for rendering. However, it is *not* a solution for light direction or light distribution. Rather, it can be thought of as an estimate of the brightness of a uniform sphere, having the same reflectance properties as the query (not necessarily Lambertian), under the same illumination as existed for the query. This is essentially a convolution with the light distribution; recovering the light distribution itself would require a costly *de*convolution step, as well as the need to make additional assumptions about the form of B_L, e.g. the assumption that it is Lambertian. The main contribution of the lightsphere algorithm is to use this convolved brightness information, which is sufficient for rendering, without requiring the costly solution for lighting, or other relatively costly computations such as least squares solutions.

$\bar{B}_L$ is an approximation to some function on the Gaussian sphere that can be used for rendering. We have used binning and averaging to estimate this function, and lookup and interpolation to evaluate it. This choice allows for a very fast algorithm. However, clearly other methods can be used to construct such a function; e.g., one could fit a spline or some other basis. This would be expected to improve accuracy results with a tradeoff of slower speed; we are pursuing some such methods in a continuation of the work reported here.

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 Eq. (3) 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 per-

[†]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.

[††]This observation is due to David W. Jacobs.

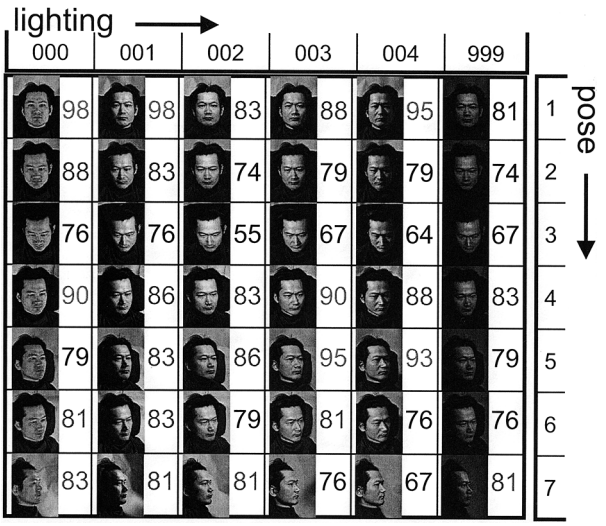

Fig. 1 Percent recognition accuracy as a function of lighting and pose of the query. Images shown are reduced monochrome versions of full color query images for one subject. The query set consists of pictures of the same 42 individuals as used in the 3D model set. (The strange numbering for lighting and pose is for historical reasons.)

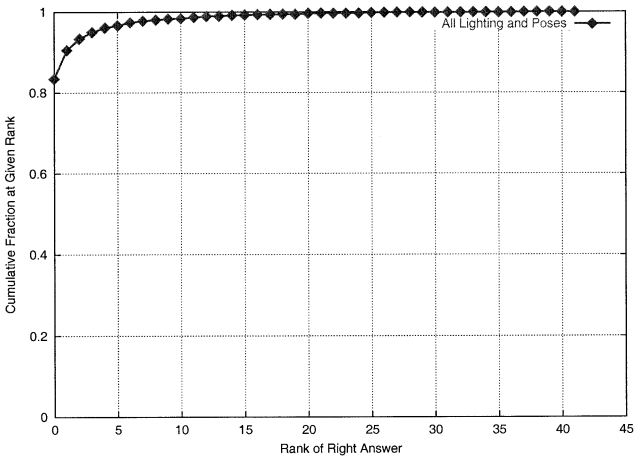

Fig. 2 Cumulative rank curve taken over all 42 combinations of pose and lighting conditions. The ordinate represents the fraction of the time that the correct answer was ranked at or above the value on the abscissa. (Rank numbering starts at 0, not at 1.)

fectly diffuse lighting. However, in practice the lighting is not perfectly diffuse. Nevertheless, to the extent that the observed intensity obeys the relation of Eq. (3), 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 Eq. (3), 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.

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. The database consists of members of NEC CRL (located in Japan), and therefore is very homogeneous in terms of race, age, and sex. (A more heterogeneous data set can be expected to result in better differentiation among individuals, and we are currently working on developing such a data set.)

The 3D model data consists of 24 bit RGB texture information on a 640×640 grid with depth values accurate to about 0.5 mm (standard deviation of error). The query data is 640×480, also 24 bit RGB. The results we present include a preprocessing step of smoothing the computed normals of each 3D model by a factor of 16×16.

In this database, query images were taken nearly at the same time as the 3D models were captured. Therefore, to a good approximation the faces behave as rigid objects, and there is little variation in facial expression. We are not attempting in this work to devise a method robust against changes in facial expression, but such work, we believe, can be built upon this work.

Below are results using semi-automatically determined poses, i.e. poses that were automatically computed from 12 or fewer fiducial points manually clicked once and for all on each model, and once on a single lighting condition of each pose of each query. We obtained the recognition accuracies shown in Figs. 1 and 2.

Figures 3 and 4 illustrate the operation of the algorithm. Figure 3 shows the sequence of intermediate results in attempting to test a 2D query against the 3D model which happens to be the correct answer for that query.

The first frame (a) shows the query image.

The algorithm is given a pose for the 3D model we want to test against that query. This pose has been computed specifically for this query-model pair, and a different pose would be computed for a different model to test. The next frame (b) shows the results of rendering the 3D candidate model in the pose that was supplied. This is a rendering only of the texture-mapped albedo; no lighting is being used, or equivalently, the lighting for this rendering is perfectly diffuse. The black area indicates data that is not used.

The next frame (c) shows a representation of the registered normals for the 3D model in that pose. (Actually this image corresponds to a color coding of 2 components of the normal; another image is needed to define the complete normal.)

Based on the query (a), rendered model albedo (b), and registered normals (c), the lightsphere data

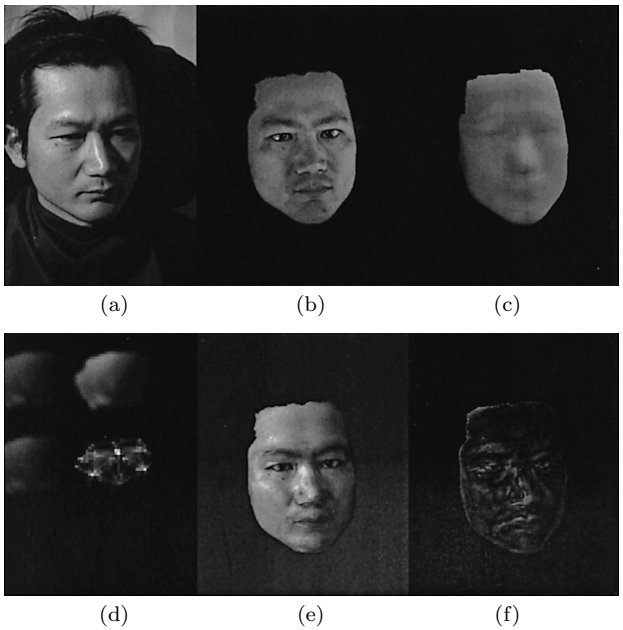

Fig. 3 Example of operation of Lightsphere for correct match, i.e. when the model being examined is that of the individual in the query. From left to right: **(a)** query image; **(b)** 3D model texture projected for previously found pose for this query/model pair; **(c)** one component of the surface 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.

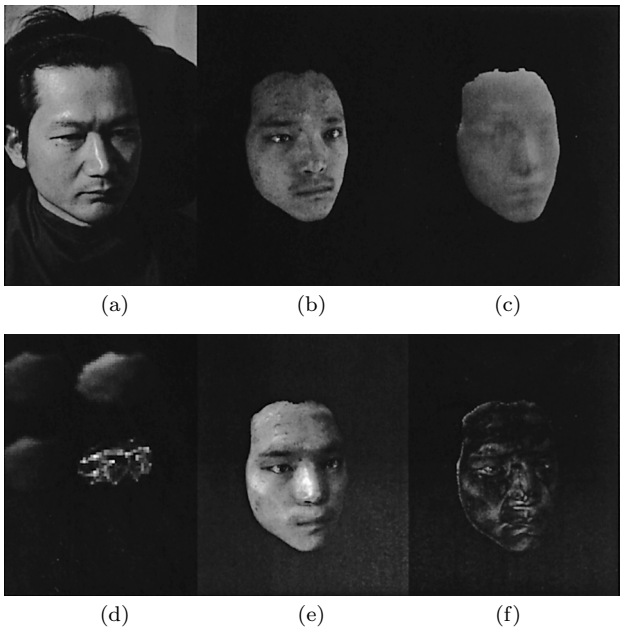

Fig. 4 Example of operation of Lightsphere for wrong match — the model is of a different individual than the query. The query is the same as that in Fig. 3, but a different model is being used. See Fig. 3 for an explanation of subparts. Note the significantly larger error than in Fig. 3 (which may reproduce 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.

structure expressing a lookup table for "brightness" as a function of normal is computed. This is displayed in the next frame (d). The first 3 subimages correspond to the 3 color channels (R, G, B), and the 4th subimage represents the sample count in each bin of the lookup table. The bin value is shown as a function of the x and y components of the unit normal only. (The z value is constrained by $x^2 + y^2 + z^2 = 1$; i.e., one can imagine looking down at a hemispherical cap painted with the lightsphere brightness coefficient.)

Using this lookup table (d), the 3D model can be rendered to conform to the light of the query. This is done by using the information in frames (b), (c), and (d) to render with lighting in the next frame (e). We emphasize that the "lighting" used for this rendering, is not actual lighting — rather, the rendering is done based on the "brightness" function computed in the lightsphere algorithm. This amounts to rendering (b) with the lighting of (a), but without knowing or solving for the actual lighting.

Finally, the lighting-rendered model in (e) is compared with the query (a), for the masked area of interest, to yield a difference image, shown in the final frame (f). The mean square error of the difference image is used as the quality measure for recognition.

Figure 4 shows the same process in the case where a comparison is being made to a 3D model that does *not* correspond to the query, i.e., for a model which is the wrong answer. In this case the difference image results in a higher mean square error, and the wrong match is rejected in favor of the correct one.

It is difficult to compare these results with other work, and any such comparisons should be viewed with suspicion. First of all, previous work does not use 3D models. Second, it is difficult to find work which uses such a wide variation of lighting and pose. Most work dealing with lighting variation uses smaller test sets. And, finally, comparisons are not meaningful unless the methods being compared used the same training data, and the same test data. In addition, the accuracy results we present here are a preliminary proof-of-concept. Since the time that we obtained these results, we have developed methods that improve the pose solution with dramatic improvement of accuracy. Lightsphere should be thought of as the lighting compensation part of a system, while recognition accuracy measures the performance of the whole system, and therefore makes it difficult to draw tight conclusions about just a single component.

As a rough comparison, we indicate some results from other work[†]. Results reported in [8] include a

[†]To derive these numbers we have made estimates, inferences, and approximations based on the information in the source. We apologize to other authors if we have misrepresented their results.

number of systems. About 4 pose/lighting conditions seem to be comparable to ours in this data set. The best reported accuracy results are, for conditions roughly comparable to our pose/lighting combinations, (1, 000): 96%, (1, 002): 82%, (4, 000): 63%, (6, 000): 5%. (Note that the 5% figure is an unfair comparison because training data was not appropriate.) These figures are for about 1000 models. Because [8] provides rank curves, we can estimate that the accuracy on a dataset of the same size as ours, 42 models, would be about (1, 000): 99%, (1, 002): 96%, (4, 000): 72%, (6, 000): 25%.

Among the best results under comparable conditions have been achieved by Georghiades, et al. [4]. For roughly comparable pose/lighting conditions their recognition accuracy is: (1, 000): 100%, (1, 002): 100%, (1, 003): 100%, (1, 004): 100 (2, 000): 100%, (2, 002): 100%, (2, 003): 99%, (2, 004): 95%, (4, 000): 99%, (4, 002): 99%, (4, 003): 99%, (4, 004): 91%. However, these results were obtained with only 10 models. Using the rank curves for lightsphere, we can estimate what the accuracy would be for 10 models: in all the cases listed for [4], lightsphere accuracy for 10 models would be 100%. Indeed, it would be 100% on nearly all combinations of lighting and pose in our experiments, if the dataset included only 10 models. (100% accuracy on 10 models is equivalent to ranking the correct model in the top 1/10 of all models, so this can readily be estimated from rank curves obtained with a larger number of models and queries.)

Acknowledgments

We are deeply indebted to Johji Tajima, Shizuo Sakamoto, and Rui Ishiyama of the NEC C&C Media Research Laboratory for generating and providing the data used in these experiments, as well as for many informative discussions that provided the insight and inspiration for the ideas presented here. David Jacobs of the NEC Research Institute (NECI) 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 NECI encouraged and made possible the collaboration with our Japanese colleagues whose fruits are reported here. Penio Penev of NECI suggested the use of vector quantization for binning, and was instrumental in running experiments and analyzing data. Jeffrey Mark Siskind of NECI graciously provided computational resources. Rui Ishiyama performed the herculean task of the manual component of the pose solutions used in our experiments. Youssef Ibrahim of Université de Montréal ran the experiments to gauge performance versus resolution. We thank Shizuo Sakamoto and Rui Ishiyama for the kind permission to use their likenesses in the figures. We also thank the reviewers of this paper for their helpful comments.

References

[1] A.P. Blicher and S. Roy, "Fast lighting/rendering solution for matching a 2D image to a database of 3D models: 'Lightsphere'," Proc. IAPR Workshop on Machine Vision Applications (MVA2000), pp.481–484, Tokyo, 2000.

[2] P.N. Belhumeur and D.J. Kriegman, "What is the set of images of an object under all possible lighting conditions?" IEEE Conf. CVPR, pp.270–277, 1996.

[3] P.N. Belhumeur and D. Jacobs, "Comparing images under variable illumination," Proc. IEEE Conf. CVPR, pp.610–617, Santa Barbara, 1998.

[4] A.S. Georghiades, P.N. Belhumeur, and D. Kriegman, "From few to many: Generative models for recognition under variable pose and illumination," Proc. 4th IEEE International Conf. Automatic Face and Gesture Recognition, pp.277–284, Grenoble, 2000.

[5] A.S. Georghiades, D. Kriegman, and P.N. Belhumeur, "Illumination cones for recognition under variable lighting: Faces," Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp.52–59, Santa Barbara, 1998.

[6] R. Ishiyama, S. Sakamoto, and J. Tajima, "A new face-recognition system with robustness against illumination changes," Proc. IAPR Workshop on Machine Vision Applications (MVA2000), pp.127–131, Tokyo, 2000,

[7] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," IEEE Trans. PAMI, vol.12, no.1, pp.103–108, Jan. 1990.

[8] P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," IEEE Trans. PAMI, vol.22, no.10, pp.1090–1104, Oct. 2000.

[9] L. Sirovich and M. Kirby, "Low dimensional procedure for the characterization of human faces," J. Optical Society of America-A, vol.4, no.3, pp.519–524, 1987.

[10] M. Turk and A.P. Pentland, "Eigenfaces for recognition," J. Cognitive Neuroscience, vol.3, no.1, pp.71–96, 1991. Earlier: "Face recognition using eigenfaces," IEEE Conf. CVPR, pp.586–591, 1991.

[11] T. Vetter, "Synthesis of novel views from a single face image," International J. Computer Vision, vol.28, no.2, pp.103–116, June/July 1998.

[12] P.A. Viola and W.M. Wells III, "Alignment by maximization of mutual information," International J. Computer Vision (IJCV), vol.24, no.2, pp.137–154, Sept. 1997.

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