Point Multiplication and Soft Caching: An Approach for Accelerating Calculations in Graphics

Mark Meyer John Anderson
Pixar Animation Studios

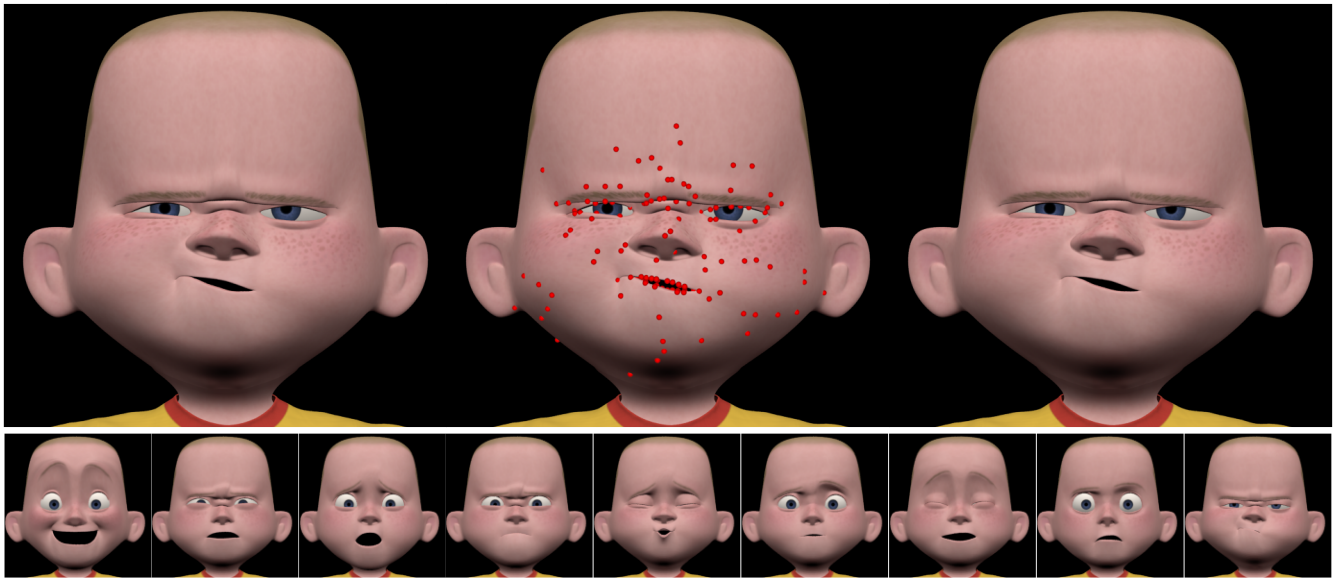

Figure 1: *An example of the point multiplication technique applied to character animation: (Top) Left: The facial model computed by posing all 2986 articulated facial points using our in house posing engine. Middle: The 170 key points computed by our Point Multiplication Algorithm. Right: The facial model computed by posing only the 170 key points using our in house posing engine and then using Point Multiplication to generate the 2986 facial points. Using the Point Multiplication technique results in an 8.7x speedup in posing time. (Bottom) Several different poses generated by our Point Multiplication algorithm are displayed to demonstrate the range of poses possible using the Point Multiplication technique.*

Abstract

Point multiplication is a statistical acceleration scheme which first identifies a set of characteristic key points in a graphical calculation such as articulation or rendering. Samples of the calculation at the key points are then used to provide a subspace based estimate of the entire calculation. Point multiplication is a useful acceleration scheme when the calculation requirements for evaluating the key point values are substantially lower than for evaluating the full set of points. Calculations of this sort occur in many areas of graphics - here we present examples from the calculation of facial articulation and the rendering of scenes with global illumination.

The soft caching process is an extension to the point multiplication technique where the key points are also used to provide a confidence estimate for the point multiplication result. In frames with high anticipated error the calculation will then "fail through" to a full evaluation of those points (a cache miss), while frames with low error can use the accelerated statistical evaluation (a cache hit).

Keywords: Animation, Subspace Analysis, Statistical Models

1 Introduction

Recently there has been a great deal of interest in the use of statistical models for calculations in graphics. One line of research illustrated by Lewis [Lewis et al. 2000] and Wang and Phillips [Wang and Phillips 2002] involves the creation of a kinematic articulation model which is trained from poses which can be generated from either physical simulations or hand corrected posed models. The Lewis approach is based on a pose based interpolation scheme in animation variables while Wang and Phillips base their approximation on multiple coefficient weighting of the positions of points in various skeletal articulation frames. Both of these approaches are much more general than the techniques described here since they do not require the posing of key points, a limitation that precludes the use of models depending upon history based simulation for posing.

The weakness of the kinematic schemes is related to their generality - the required training sets for these methods can be very large and it is essentially impossible to place bounds on the errors of the reconstructions when poses are seen which are far from those included in the training set.

Our technique is most similar to EigenSkin [Kry et al. 2002] and Precomputed Local Radiance Transfer (PLRT) [Kristensen et al. 2005]. EigenSkin is a technique for real time deformation of character skin. EigenSkin uses principal component analysis to compute a basis for the skin displacements resulting from perturbing each joint of a model. Given several example poses consisting of

joint angles and skin displacements, specific joint angle values can be associated with coordinates in the basis space by projecting the skin displacement into the basis. Skin displacements for novel joint angle configurations are computed by using the joint angles to interpolate these example basis coordinates, and the resulting coordinates determine a displacement represented in the subspace formed by the basis.

Precomputed Local Radiance Transfer (PLRT) is a method for accelerating the computation of illumination in a scene. First, the illumination is computed for several example configurations (light positions, etc.) the resulting illumination is stored at each vertex as the coefficients to a spherical harmonic basis. Clustered PCA is then performed on these coefficients to create a subspace and the subspace coordinates (projection onto the basis vectors of the subspace) of the example configurations are computed. The illumination from a novel lighting position can be computed by using the light position to interpolate the subspace coordinates of nearby example lights, and the resulting subspace coordinates determine the illumination represented in the subspace.

Our Point Multiplication scheme is similar to these techniques in that it computes a subspace from examples and uses this subspace model to accelerate the corresponding calculation. Where our technique differs from EigenSkin and PLRT is that instead of using animation variables such as joint angles and light positions to drive the subspace model, we instead use the samples of the calculation itself (skin displacements or illumination, here) at a set of carefully chosen key points to drive the projection.

The Point Multiplication / Soft Caching (PMSC) strategy is particularly applicable to rendering with global illumination. When applied to a photon mapping renderer [Jensen 1996] the illumination gather step is initially limited to only the key points. If error tolerances are exceeded the fail through is easily accommodated through additional gathers.

Most of the current approaches to accelerating global illumination involve the use of importance sampling and spatial filtering operations for noise reduction. The PMSC approach can be interpreted in this light. The point multiplication stage can be thought of as a set of optimal spatial filter kernels derived following the Capon [Capon et al. 1967] formalism and the key point positions can be interpreted in an importance sampling context. In this context the contribution of the PMSC technique is to add the introduction of prior knowledge about the light transport within the scene geometry, as observed from the training set, to optimize the spatial filtering and importance sampling operations.

2 Overview

A schematic block diagram of the general statistical modeling problem is shown in Figure 2. In this general model we have a core black box which we would like to train to accept the animation variables (posing controls, light positions and properties) and produce the desired outputs (point positions or illumination values). The problem with this most general form is that the relationships between the input controls and the outputs can be highly nonlinear - for example it is common to use an articulation variable to set either the sensitivity or pivot point of another variable - an extremely nonlinear process which is essentially impossible to discover statistically from a training set.

Various approaches to this problem have been tried. Lewis [Lewis et al. 2000] has proposed a pose interpolation method where one

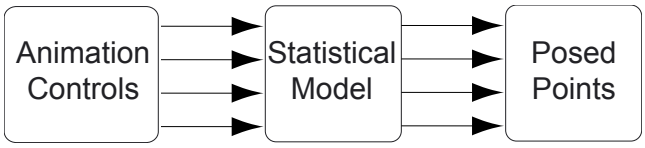

Figure 2: *Block Diagram of a basic statistical modeling system for animation: The trained statistical model accepts as input the animation variables and produces the posed points as output. Unfortunately, since the relationships between the animation variables and the posed points can be extremely nonlinear, both the statistical model as well as the training set must account for these complex nonlinearities.*

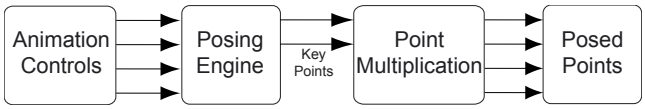

Figure 3: *Block Diagram of the point multiplication system applied to animation: Rather than taking the animation controls directly as input, point multiplication instead computes the positions of a small set of key points using the standard posing engine and then takes these posed key points as input. Point Multiplication then uses the values of these key points to produce all of the posed points. The advantage here is that if the key points are chosen such that the behavior of these key points describes all of the important nonlinearities of the problem, the statistical model can compute the remaining point values without having to deal with these nonlinearities.*

looks for nearby training poses in the high order pose space defined in terms of "locally important" animation variables. Wang and Phillips [Wang and Phillips 2002] approach the problem of describing skeletal articulation with a model based on linear tensor products of point positions represented in multiple rotated coordinate frames.

The approximation used in the PMSC approach is quite simple and is depicted in Figure 3. A training set is used to create a statistical subspace model that is driven by runtime computed values of a set of key points. We have chosen to use a model that is linear in the key point values. Thus we must choose our key points from the training set such that the behavior of the key points describes all of the important nonlinearities of the problem. It is of course quite possible that there may be new nonlinearities that are not seen in the training set and that is where the need for a soft cache fail through arises. In the rest of this paper we will discuss the two key issues: how we select the subspace and key points, and how we provide error estimates to be used in the fail through process.

3 Animation and Rendering Subspaces

The use of subspace descriptions to reduce the number of degrees of freedom in a process is quite common and goes back at least as far as [Lorenz 1956] who developed the Empirical Orthogonal Function (EOF) method in an attempt to classify weather patterns for use in statistical weather forecasting. Lorenz realized that singular value decomposition could be used to provide a least squares optimal decomposition into a low dimensional subspace, and that the coordinates of this projection could be used for classification. An extension of this technology to facial animation was developed in the "Making Faces" [Guenter et al. 98] system that demonstrated the use of a low dimensional pose space derived from high resolution motion capture data.

Following the Lorenz formulation we start with a set of training frames of a state variable $\mathbf{Q}$, usually a position or illumination value, that are intended to exercise the pose space of the system to be accelerated. We then compute a set of basis functions $\mathbf{a}_i(x)$ and $\mathbf{b}_i(t)$ where x is a generalized "space" coordinate which identifies the point and t is a "time" coordinate indexing the training frames:

$$\hat{\mathbf{Q}}(x,t) = \sum_{i=1}^{M} \mathbf{a}_i(x)\mathbf{b}_i(t).$$

Here $\hat{\mathbf{Q}}$ is an M dimensional subspace approximation to $\mathbf{Q}$.

Since, in our applications, x and t are discrete variables, it is notationally convenient to rewrite the previous equation as:

$$\hat{\mathbf{Q}}[x,t] = \sum_{i=1}^{M} \mathbf{a}_i[x]\ \mathbf{b}_i[t].$$

where $\hat{\mathbf{Q}}$ (similarly $\mathbf{Q}$) is a matrix indexed by x and t, $\mathbf{a}$ is a vector indexed by x, and $\mathbf{b}$ is a vector indexed by t.

Following the EOF (and the closely related singular value decomposition (SVD)) technique we can find the choices for the $\mathbf{a}$'s and $\mathbf{b}$'s which minimize the least squares errors for all values of M from by finding the eigenvectors of the two covariance matrices [1] $\mathbf{Q}\mathbf{Q}^T$ and $\mathbf{Q}^T\mathbf{Q}$. These two matrices have the same set of eigenvalues up to the minimum of the number of x's and t's. $\mathbf{a}_1[x]$ and $\mathbf{b}_1[t]$ are the eigenvectors with the largest eigenvalues, $\mathbf{a}_2[x]$ is associated with the second largest eigenvalue, etc. In our usage we will refer to the $\mathbf{a}$'s as the subspace basis vectors and the $\mathbf{b}$'s as the pose space coordinates

This particular decomposition has a number of important properties. It is sequential, that is if we perform the process for $M = 1$ we will get the first set of vectors. Continuing this process by performing the same analysis on the residual data, $\mathbf{Q} - \hat{\mathbf{Q}}$ we will get the second set of vectors and so on. This iterative process is actually one of the numerically attractive methods for computing the subspace since the largest mode can be easily found from a factored power method without computing the eigenvalues or even computing the covariance matrices [Jalickee and Klepczynski 1977]. Since the process can be iteratively performed one can monitor the magnitude of the residual, which is the subspace projection error, and terminate the process when the error is acceptably small.

A significant aspect of this technique is that at no point in the process have we made any assumptions about the spatial or temporal adjacency of points and frames. The $\mathbf{a}_i[x]$ vectors do tend to be smooth in space but this smoothness results only from the smoothness and spatial correlations of the original training frames. In fact one interpretation of the $\mathbf{a}_i[x]$ vectors is that of optimal filter kernels for the detection of pose space coordinates in the presence of noise [Capon et al. 1967].

It should also be noted that once the final subspace dimension, M, is chosen, the $\mathbf{a}_i[x]$ and $\mathbf{b}_i[x]$ vectors are not uniquely determined by the minimization of error. The only critical property is the subspace spanned by the basis vectors. In fact the multiplication by any nonsingular matrix will result in vectors which span the same space and the multiplication by any orthogonal M dimensional rotation matrix will result in an orthonormal basis for the subspace. This property is quite useful and is frequently exploited in the factor analysis community [Harmon 1967] to generate basis vectors that are more "local" in some sense than the original basis vectors. We will take advantage of this when we select the key points.

In these terms the point multiplication algorithm can be stated quite simply: after computing a subspace and set of key points using the training set, we take the values of selected key points and use a least squares projection onto the subspace to derive the pose space coordinates. The statistically determined pose is then taken as the value of $\hat{\mathbf{Q}}$ for that pose space coordinate. All that remains is to determine how to select the key points.

4 Selection of Key Points

When using key points to locate a pose in a subspace there are two potential sources of error. The first of these is the projection error that results from the fact that the pose itself may not be in the subspace. The second is the cueing error - the error that results from the fact that the subspace location determined from the least squares fit to the key points may not be the closest point in the subspace to the desired pose due to the deficiency of our key point values as distance proxies. We have derived an iterative approach for selecting the key points that attempts to minimize this cueing error.

Our approach starts by generating a block of basis vectors, typically 10 to 20. Next we perform a coordinate rotation on the basis vectors. We use the Varimax method [Harmon 1967] which computes an orthogonal rotation that maximizes the ratio of the 4th moment to the 2nd moment of the basis vectors, a measure of locality. Intuitively, this rotation localizes the basis vectors by maximizing the variation of a small set of points in each basis vector (in turn driving the variation of the other points towards zero).

Once we have a set of suitably conditioned basis vectors we choose two points from each vector, the first point is the point with the largest magnitude in the basis vector and the second is the point whose inner product with our first point has the largest negative value. This type of technique is used in statistical climatology to discover statistical structure and is known as teleconnection analysis. This technique is essentially choosing the point that moves the most when this (localized) basis function is excited as well as the most negatively correlated point.

In addition to the points computed by the above procedure, there are often critical fiducial points identified by the character (or lighting) designer that are seeded into the first batch. For rendering applications we may also add a sparse grid of points to ensure that no large regions are devoid of key points.

Given this set of key points and the subspace basis vectors, we can compute a point multiplication approximation for each of the columns of the training matrix $\mathbf{Q}$. Subtracting these approximations from the corresponding columns of $\mathbf{Q}$ creates a residual matrix which represents the error (both projection and cueing error) when approximating the training set using the current set of key points. We then repeat the key point selection process to add additional key points, using the residual matrix to compute the basis vectors. Using the residual in this iterative process allows the key point selection to choose points to reduce the cueing error.

The process terminates when the maximum residuals reach an acceptable error bound. A subtle but important point is that the basis vectors constructed during this phase in the second and later batches should now be discarded - all basis vectors used for the runtime point multiplication reconstruction should be computed from the initial training matrix $\mathbf{Q}$.

[1] Without loss of generality, $\mathbf{Q}$ is assumed to have zero mean

The process of selecting key points can be summarized as follows:

keyPoints = initial user defined key points
resid = **Q**
Iterate until **resid** is small:
  Compute basis vectors from **resid**
  Varimax rotate the basis vectors
  Choose the control points from each rotated basis vector (excluding points that are already in keyPoints) and add them to keyPoints
  Reconstruct the columns of **Q** using point multiplication with the current set of key points, call this $\hat{\mathbf{Q}}$
  resid = $\mathbf{Q} - \hat{\mathbf{Q}}$

5 Soft Caching and Fail Through

Although estimates produced by the Point Multiplication technique are usually quite accurate, there can be times when the statistical reconstruction is unable to produce an estimate with acceptable accuracy. An example of this is shown in Figure 4 for the application of facial animation. The facial pose resulting from posing all of the points with our posing engine is shown on the left, while the result generated via Point Multiplication of 170 key points is shown in the middle. Notice how the tight lip pucker is incorrectly reconstructed.

Remember that the first step in our point multiplication process involves the least squares calculation of a set of subspace coordinates from the key points. If the key points are well chosen, the error in this calculation is a good measure of the total projection error. Large projection errors occur in cases when we are calculating the results for a frame where an animation control is exercised that was not exercised in the training set or a particular combination of animation controls is exercised that result in a novel pose.

This suggests the possibility that we could treat the Point Multiplication result like a cache and use a large key point projection error as an indication of a "cache miss". Since the domain of applicability of our method involves calculations easily performed on a point by point basis, in the event of a cache miss, we could simply go on to perform the full calculation on all of the points. For animated sequences somewhat more subtlety is required since we do not want to have any discontinuous behavior that will lead to popping. To eliminate this problem we incorporate a transition zone of projection error values $\varepsilon = [\varepsilon_{min}, \varepsilon_{max}]$ over which we do perform the full calculation but interpolate between the point multiplication result and the full calculation result as a function of error. Specifically, if the error (measured by RMS error of the key points) is below ε_{min}, only the Point Multiplication result is used, if the error is greater than ε_{max}, the full calculation is used at every point, and if the error is in the ε range, the result is linearly interpolated between the Point Multiplication and fully calculated results.

Figure 4(right) shows the result of applying Soft Caching to the incorrectly reconstructed facial pose in Figure 4(middle) using $\varepsilon = [0.1, 0.15]$. Notice how the cache miss was correctly identified and the resulting pose is much more accurate. We should also mention that poses identified as cache misses can be tagged and used as additional training for the PMSC system. This results in a system that learns more about the pose space as it is used. Training could be done each night resulting in a more accurate PMSC system for the animators and lighters to use the next day.

6 The Complete PMSC Algorithm

Now that we have described all of the pieces, the complete PMSC algorithm can be formulated as follows:

Preprocess
  1) Compute the basis vectors $\mathbf{a}_i[x]$ from the training set **Q** (Section 3)
  2) Compute the key points (Section 4)
Runtime
  1) Compute values only at the key points
  2) Least squares project the key points onto the subspace to find the pose space coordinates $\mathbf{b}_i$
  3) Reconstruct all points using $\mathbf{b}_i$:
    $\hat{\mathbf{Q}}[x] = \sum \mathbf{a}_i[x]\ \ \mathbf{b}_i$
  4) Soft Cache (Section 5):
    Using the projection error of the key points and the user defined ε determine if a cache miss has occurred. If it has, compute the full solution at all points and interpolate with the point multiplication solution.

Note that the runtime component of the PM algorithm is extremely cheap (compared to the full solution), only requiring the computation of the key point values, a small least squares projection and a linear combination of the basis vectors.

7 Applications to Character Articulation

The first target domain of our study was in the area of non-skeletal character articulation, particularly facial articulation. Faces are particularly difficult to replace with statistical models because they often have hundreds of animation controls, many of which are very complex and non-linear. On the other hand the results of Guenter *et al.*'s [Guenter et al. 98] work have shown that, at least for human faces, the number of real degrees of freedom is often quite small. This suggests that a subspace based technique should work well for faces. Unfortunately, using the animation variables directly to drive the subspace model is difficult due to the complex nonlinearities. Additionally, faces do not have the benefit of skeletal structure and joint angles used by other parts of the body as input to their statistical models. By using the computed key point positions as input, our Point Multiplication technique is able to avoid modeling these complex nonlinearities and performs well when applied to facial articulation.

We were very fortunate to be able to get access to the facial animation for a boy character from an entire CG feature film for testing purposes. The data set consists of about 8322 frames of animation from 137 animation sequences. In order to provide an independent data test for the point multiplication process we divided the animation sequences into two sets, one was used to train the model (2503 poses chosen from 44 randomly selected sequences) and the second was used for validation (the remaining 5819 poses). We found that 85 basis vectors posed by 170 key points was sufficient to pose the model to very small errors in the training segments. The character with the key points indicated is shown in the top-middle of Figure 1.

One important question related to facial articulation is how much acceleration is possible. On our characters, which employ fairly complicated kinematic deformer networks, we have found that faces lie in between the near linear cost per point calculation and a constant cost (where posing a small set of points is no less expensive than posing than the full face). For our test character (Figure 1), we have found that the cost of posing our 170 key points is about 10% of the cost of posing the full 2986 points. In particular, all 2986 points can be posed in 0.5 seconds on average, while the 170 key points can be posed in 0.05 seconds on average. The Point Multiplication itself takes 0.00745 seconds on average resulting in an 8.7x speedup in posing time. We do not achieve a speedup linear with the key point to total point ratio for a few reasons: First, the key points often require more complicated deformations than the other points. For example, the mouth points require more effort

Figure 4: *An example of Soft Caching applied to character animation: (left) The face generated by posing all 2986 points using our in house posing engine. (middle) The pose generated using point multiplication of 170 key points. Notice how the lips are improperly reconstructed. (right) That pose generated using Soft Caching with* $\varepsilon = [0.1, 0.15]$. *Notice that the pose generated by Point Multiplication was correctly identified as inaccurate using the key point error and the cache miss fail through correctly interpolated the Point Multiplication pose with the fully computed pose producing much more accurate lips.*

to pose than the points on the back of the head. Second, multiple points can sometimes reuse computations - so posing twice the points won't necessarily require twice the computations.

In order to validate our technique, we have examined the Point Multiplication results using the independent test poses. Figure 1 shows an example of the quality of the Point Multiplication result. The top-left image shows the character with all 2986 points posed using our in house posing engine, while the top-right shows the result of posing only the 170 key points with our in house posing engine and then using Point Multiplication to generate the 2986 facial points. Notice that the Point Multiplication result is an extremely accurate recreation of the fully computed pose. The accompanying video shows the side by side results for the more than the first 800 poses in the independent test set. As can be seen in the video, the Point Multiplication results match the fully computed results very well - the maximum error for any point over 99.5% of the test poses is less than 1% of the diagonal of the head bounding box.

For most of the test poses the Point Multiplication result and the fully computed result are visually similar and any differences are only noticeable though detailed examination of the side by side results. However, as noted in section 5, there are times when the estimated pose does not accurately match the fully computed pose. Figure 4 and the video show an example of one of the worst poses in our test set. The left image displays the pose generated by full computation, while the middle shows the result of Point Multiplication. Notice how the lips in particular do not match between the two techniques. In these cases, the Soft Caching mechanism (section 5) can detect and fix such poses. Using $\varepsilon = [0.1, 0.15]$, the Soft Cache detects the incorrect poses and generates a more accurate pose as can be seen on the rightmost image and in the video. With these settings, we have a cache hit rate of 96.6% for the entire dataset (8042 poses out of 8322 poses). Note that poses that cause a cache miss do pose slightly slower than simply computing a full pose (due to the subspace projection, error computation and possible interpolation), however it has been less than 5% slower in all our tests.

Although our test character used a kinematic deformation rig, it should be noted that interesting effects can be achieved for characters using poses defined by physical simulation or hand sculpting. First we define a version of the face that can be manipulated by the animator at runtime (the control face). Then we create a training set containing pairs of the physically simulated (or hand sculpted) face and the corresponding pose of the control face. At runtime, the animator poses the control face and the system uses the control face points to find the projection into the joint (control face, simulated face) subspace and computes the corresponding simulated face. Note that in this "Cross Point Multiplication" case the Soft Cache fail through to a full calculation is not possible.

8 Applications to Rendering

The second problem domain we explored was the computation of indirect illumination. These rendering problems are particularly well suited to acceleration using our PMSC approach since they often have a large fraction of the total computation in a final gather step whose computational costs vary nearly linearly with the number of points computed.

We have constructed a test case for the indirect illumination problem based on the Cornell box. Each training frame was generated by illuminating the box using a single point light from a grid of 144 lights along the ceiling (see Figure 5(a)). Training results in 32 illumination basis functions and 200 control points as seen in Figure 5(a). We have validated the results by computing the indirect illumination at a series of light locations scattered near the training array. Figures 5(b), (c) and (d) shows comparisons of the fully computed lighting versus the lighting computed via Point Multiplication for a few select frames. As these images show, the lighting is accurately estimated and the resulting errors are quite small.

Note that the use of the lighting at key points as input to our system as opposed to lighting variables (such as light position) allows our system to handle changes in the lighting variables rather easily. For instance, changing the light type, or distance falloff, etc. will have a complex, nonlinear effect on the resulting indirect illumination. Since our system actually computes the indirect illumination at the key points and uses the illumination itself to drive the statistical model, we do not require a way to map each of these animation variables to the final indirect illumination as other methods would. In a production setting where lights have many complex and often interacting controls, this is huge benefit. We are in the process of

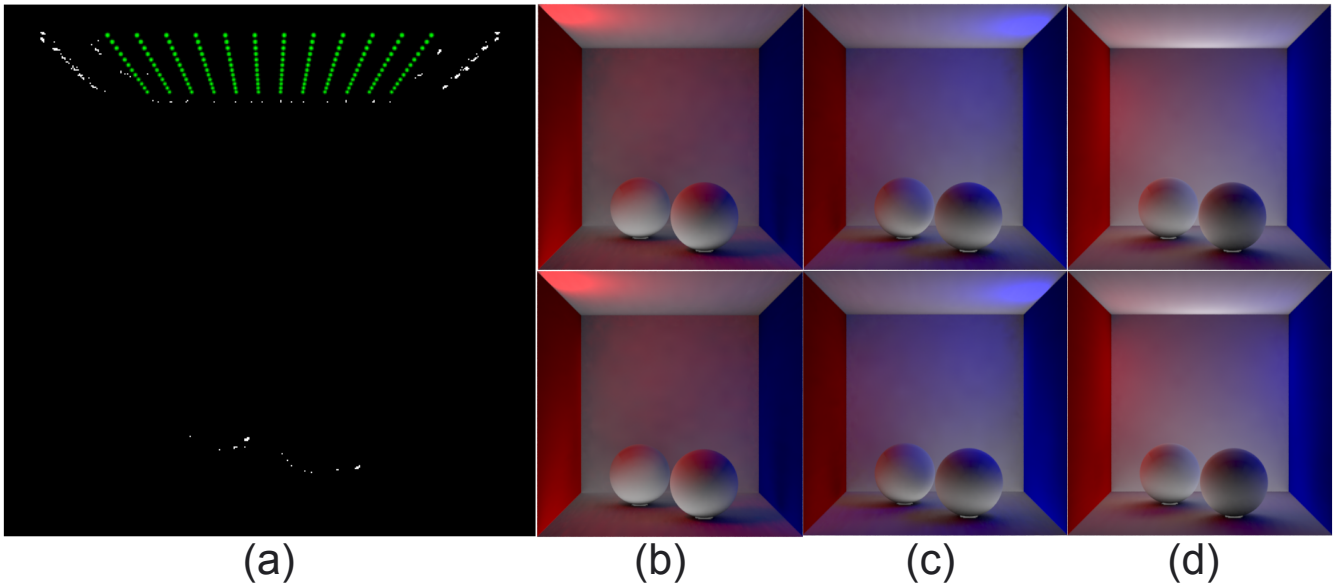

Figure 5: *An example of Point Multiplication applied to rendering indirect illumination: (a) The green points show the locations of the point lights used to generate the training set and the white dots are the 200 key point locations computed by the the Point Multiplication technique. (b,c,d) Comparisons of the fully computed solution using 160000 points (top) to the Point Multiplication solution computed from the 200 key point values (bottom) for novel light positions. Note that the Point Multiplication results are extremely accurate.*

exploring the use of this sort of technique in interactive relighting including global illumination.[2]

For the rendering tests presented here we have performed the calculations using image space locations, however the illumination data can easily be stored in a spatial data structure or texture map to accommodate moving cameras and geometry.

9 Summary and Ongoing Work

In this paper we have presented a methodology for accelerating certain classes of calculations that are common in computer graphics. This methodology involves the use of a statistical model that is linear in key points and can fail through to a full calculation in the event of a "cache miss". By using the key point values to capture the nonlinearities of the space, the statistical model can compute the additional points without having to deal with these nonlinearities.

One area of ongoing work involves "local" fail through. In general it is reasonable to expect that subspace projection error should be local, confined to a particular unexercised animation control or local rendering feature such as a contact shadow. We are currently in the process of exploring the use of various forms of basis function rotation and clustering to be able to localize the projection error so that the fail through process would only need to compute the full calculation on a subset of the domain.

References

CAPON, J., GREENFIELD, R. J., AND KOLKER, R. J. 1967. Multidimensional maximum-likelihood processing of a large aperture seismic array. *Proc. IEEE 55*, 2, 192–211.

GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. 98. Making faces. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 55–66.

HARMON, H. H. 1967. *Modern factor analysis*, 2nd ed. Chicago: University of Chicago Press.

JALICKEE, J. B., AND KLEPCZYNSKI, W. J. 1977. A method for compacting navigation tables. *Journal of The Institute of Navigation 24*, 2, 125–131.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*.

KRISTENSEN, A. W., AKENINE-MOELLER, T., AND JENSEN, H. W. 2005. Precomputed local radiance transfer for real-time lighting design. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1208–1215.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 153–159.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 165–172.

LORENZ, E. 1956. Empirical orthogonal functions and statistical weather prediction. Scientific Report No.1, Statistical Forecasting Project, MIT Dept. of Meteorology, December.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 129–138.

[2] There are of course other ways to use the spatial correlation data discovered from our training set in accelerating rendering operations. The use of the basis functions themselves as optimal filtering kernels for sampling noise reduction is the subject of a related paper [anonymous 2006].