

TOWARDS AN AUTOMATIC APPROACH FOR VIEW-DEPENDENT GEOMETRY

ROHIT JAIVANT KATE*

PREM KALRA

SUBHASHIS BANERJEE

Department of Computer Science and Engineering

Indian Institute of Technology

New Delhi 110016, India

E-mail: {pkalra,suban}@cse.iitd.ernet.in

View specific distortions of a 3D model according to its reference drawings are required for cartoon animations. Rademacher in [1] introduced a technique called View-Dependent Geometry wherein a 3D view-dependent model changes shape based on the viewing direction. The main problem in constructing a view-dependent model is of deforming the 3D base model according to its reference drawings. In this paper we propose an automatic method for solving this problem based on projective camera model. We present results demonstrating our approach.

Keywords: cartoon animation; 3D deformation; camera model.

1. Introduction

For designing 3D objects for cartoon animation, modelers begin with a set of reference drawings of the intended object from different view-points. As these drawings are typically hand-drawn they lack the geometric precision of the physical 3D space. Often artists knowingly violate the geometric precision in order to achieve best aesthetic effects. As a result the drawings do not actually correspond to the different views of a single rigid 3D model. So conventional 3D models as have been used in Computer Graphics fail to capture the desired views of the object.

Rademacher in [1] proposed a technique called View-Dependent Geometry - a geometry that changes shape of a 3D model based on the direction it is seen from. It makes view-dependency an inherent part of the model and calls it view-dependent model. A view-dependent model consists of a base model (a conventional 3D object, typically a triangular mesh) and a description of model's exact shape, called key deformation, as seen from specific view-points. These specific view-points are called key view-points. The key deformations are deformed versions of the base model with the same vertex connectivity. Rademacher in [1] also describes an interpolation technique by which appropriate deformation of the model can be generated automatically for any arbitrary view-point such that the deformation is consistent with the specified deformations for the given key view-points. A convivial interface is designed in [2] to further extend the potential of View-Dependent Geometry from

*Present address: Department of Computer Sciences, The University of Texas at Austin, USA

user’s perspective.

With a 3D base model and its reference drawings as input a view-dependent model can be constructed by creating a key deformation for each reference drawing such that the key deformation obtained looks like the reference drawing from the view-point that was intended while drawing the reference drawing. The technique used for this in [1] is largely manual. It first determines the view-point for each reference drawing (i.e. camera position and orientation relative to the base model which leads to a projection that best matches the given drawing) by rotating and translating the camera until the main features match. The second step is deforming the base model to obtain a key deformation. This was done by selecting a few vertices of the base model and dragging them till they occupy their intended positions. In the process the neighboring vertices are also dragged suitably to achieve a smooth deformation.

This method requires artistic skill from the user to be able to manually modify the 3D model to align it with the reference drawing. Also the deformation can take place only in 2D, parallel to the image plane of the camera. In the next section we propose a more automated approach for this problem. Our approach does not require any artistic skill from the user and the deformation we achieve is in 3D which is more intuitive and realistic. The problem addressed here is qualitative in nature and its accuracy depends on visual satisfaction. This is how the method was evaluated in [1]. We show results qualitatively similar to [1] using our approach.

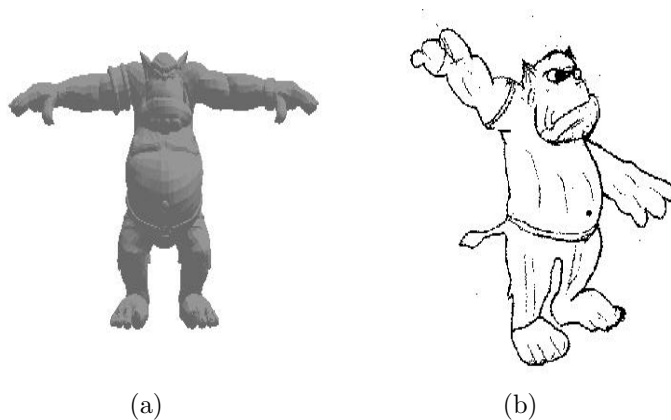


Figure 1: (a) The 3D base model, and (b) the reference drawing.

2. Approach

We are given a 3D base model and its 2D reference drawing which is the model’s desired shape as seen from the specific view point. Figure 1 gives an example, (a) shows a 3D model and (b) shows a reference drawing. Using the reference drawing

we intend to deform the 3D model so that it looks like the reference drawing from that specific view point. In this example the arms of the model need to be deformed from their horizontal positions to tilted positions.

Our approach is semi-automatic in which the only manual part is specifying corresponding points between the 2D reference drawing and the 3D model. Using these correspondences we estimate a projection matrix P . A projection matrix relates homogeneous coordinates of 3D point \bar{X} to the homogeneous coordinates of the corresponding 2D point \bar{x} after the projection, [3].

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

where $(x_1, x_2, x_3)^T$ and $(X_1, X_2, X_3, X_4)^T$ are homogeneous coordinates related to \bar{x} and \bar{X} by

$$\bar{x} = (x, y) = \left(\frac{x_1}{x_3}, \frac{x_2}{x_3} \right)$$

$$\bar{X} = (X, Y, Z) = \left(\frac{X_1}{X_4}, \frac{X_2}{X_4}, \frac{X_3}{X_4} \right)$$

The projection matrix P has eleven degrees of freedom since only the ratios of its elements are important. In the special case when the size of the object is small compared to the viewing distance one can use an affine camera model where the projection matrix has p_{31} , p_{32} and p_{33} as zeros.

The next step is deforming the 3D model. We first displace the 3D corresponding points of the model so that they occupy their required positions. Then the entire model is deformed accordingly using a 3D space interpolation technique. Note that though we are using a camera model we don't require any camera calibration.

The entire procedure is elaborated in the following sub-sections.

2.1. Finding Correspondences:

We need to establish correspondences between the selected points on the reference drawing and the points of the 3D model. This is the only step we do manually. We first project the 3D model on to an image I using any convenient projection matrix T , convenient in the sense that the desired 3D corresponding points should be visible on the image. Then for every 2D point p_i selected on the reference drawing we mark the corresponding point $q_i = (x, y)$ on the image I . As we know the projection matrix, we shoot a 3D ray through q_i which is the locus of all the 3D points which are projected on q_i . Any point (X, Y, Z) on this ray satisfies :

$$x = \frac{XT_{11} + YT_{12} + ZT_{13} + T_{14}}{(XT_{31} + YT_{32} + ZT_{33} + T_{34})}$$

$$y = \frac{XT_{21} + YT_{22} + ZT_{23} + T_{24}}{(XT_{31} + YT_{32} + ZT_{33} + T_{34})}$$

where T is the projection matrix and the homogenizing coordinate of (X, Y, Z) has been taken as 1.

Then we find the nearest intersected point r_i between this ray and the 3D model and take it to be the corresponding 3D point of p_i .

Note that we may not use a single projection matrix for the above process but may change it to make the other desired 3D corresponding points visible, for example rotating the object to obtain its back side.

Figure 2 shows the corresponding points between the reference drawing (a) and an image of the 3D base model (b). The points are marked with dots on the reference drawing and on the 3D base model. The object used here is a triangular mesh of 18,539 triangles. The corresponding point at the tip of the tail is on the back side of the model and hence not visible in (b).

For estimating the projection matrix (i.e. for determining the view point for the 3D model which aligns it with the reference drawing) we can't assume an exact alignment because artistic distortions are already present in the reference drawing. Hence for estimating the projection matrix we only try to match the undeformed features of the model with the reference drawing. For this reason, to estimate the projection matrix, we take corresponding points mostly from the undeformed portion of the model, for e.g. in figure 2 we haven't taken corresponding points from the model's arms. The features of the model which don't match exactly with the reference drawing (like the model's arms in our running example) get later deformed to match it in the deformation step. Later in the deformation step we will need to add some more corresponding points from the model's arms.

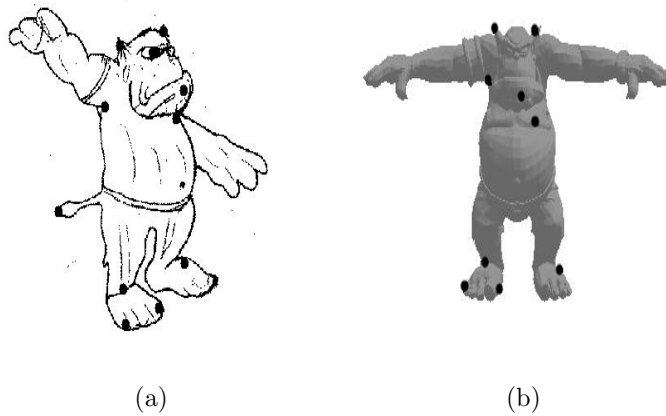


Figure 2: (a) Corresponding points on the reference drawing, and (b) corresponding points on an image of the 3D base model.

2.2. Estimating the Projection Matrix:

Using the correspondences thus obtained we estimate a projection matrix P which is approximately the projection that could have been used to obtain the 2D reference drawing from the 3D model. So P indicates the key view-point. Experimentally we have found that for the models we have used, we obtain the best results if we use the affine camera model. Thus the first three elements of the last row of the matrix are made zeros. So there are nine unknowns left to be determined. These can be obtained with nine corresponding point pairs but to make it robust we consider more points. This makes the system over-determined. Practically we have found that about 10 to 12 correspondence points work fine for estimating the projection matrix.

We can obtain a solution of an over-determined system $MX = 0$ in the unknown vector X by solving $\min_X \|MX\|^2$. This is a standard linear algebra problem whose solution is the eigenvector with minimum eigenvalue of $M^T M$. Hence we solve our system by first reducing it to the form of $MX = 0$. For every 2D point $p_i = (x_i, y_i)$ and its corresponding point $r_i = (X_i, Y_i, Z_i)$ we have the relation :

$$u_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

where u_i is the homogenizing coordinate of p_i and without loss of generality the homogenizing coordinate of r_i has been taken as 1. To reduce it to the form $MX = 0$, let $X = (P_{11}, P_{12}, P_{13}, P_{14}, P_{21}, P_{22}, P_{23}, P_{24}, P_{34})^T$ and for every point pair i generate two rows of M as shown :

$$\begin{bmatrix} & & & & - & - \\ & & & & - & - \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -y_i \\ & & & & - & - \\ & & & & - & - \end{bmatrix} X = 0$$

Figure 3 shows the model as seen from the computed projection matrix, notice the similarity of its view point with that of the reference drawing. This method lets us estimate the projection matrix automatically. It may be noted that because we aim for a key deformation which is visually satisfying, we evaluate the result of this intermediate step only by visual inspection.

2.3. Deforming the 3D model:

As was noted towards the end of section 2.1, the corresponding points used to estimate the projection matrix are mostly from the undeformed parts of the model. Hence in this step we need to add more control points from the parts of the object



Figure 3: The 3D base model as seen from the computed P .

which are to be deformed. But we also keep the previous points so that the structure of the undeformed portions of the model is preserved while the required portions get deformed. The number of corresponding points to be added depends on how much portion of the model has to be deformed. For e.g. in the running example, the model's both the arms are to be deformed, so there should be enough corresponding points to cover both its arms.

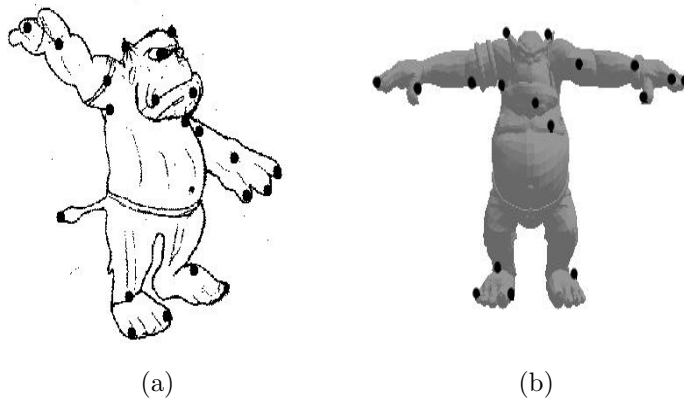


Figure 4: (a) More corresponding points on the reference drawing, (b) and more corresponding points on the image of the 3D base model.

Figure 4 shows more corresponding points taken between the reference drawing (a) and the base model (b). Here we have added eight more corresponding points which easily cover the model's arms. There is no hard and fast rule to select such control points, but practically taking just a few points which cover the portions of

the model to be deformed works well. This works because we interpolate the 3D space around the 3D corresponding points to achieve the deformation.

We call the 3D corresponding points as control points. We will first displace the control points so that they occupy their desired positions and then deform the entire model using an interpolation technique.

Using the estimated projection matrix of the 2D reference drawing, for every corresponding point p_i on the reference drawing we shoot a 3D ray. Then we displace the corresponding 3D point r_i of the 3D model such that it occupies the point s_i on the ray which is nearest from the initial position r_i . We choose the nearest points in order to achieve the deformation with least displacements of the control points from their initial positions. Note that as the control points are few and sparse, there is very little chance that this step may cause any problem like the mesh running through itself etc. We are not changing the entire mesh directly in this step, we first displace the control points and then the mesh is interpolated accordingly.

Mathematically s_i is the solution of :

$$\min_{(X,Y,Z)} (X - X_i)^2 + (Y - Y_i)^2 + (Z - Z_i)^2$$

subject to:

$$P_{11}X + P_{12}Y + P_{13}Z + P_{14} = P_{34}x_i$$

$$P_{21}X + P_{22}Y + P_{23}Z + P_{24} = P_{34}y_i$$

The solution is obtained by differentiation and substitution of variables. Now s_i will be projected to p_i as expected out of deformation.

In order to deform the 3D model according to the displaced control points, one may use any of the several existing 3D deformation techniques. For instance, global deformations like free form deformations [4], “wires” [5], scattered data interpolations [6] or other existing techniques.

We have used tetrahedralization approach ([7]) as it is simple and fast. We first tetrahedralize the 3D space using the control points (before displacement). Then for every point ρ of the model (vertices of triangles as our model is a triangular mesh) we find in which tetrahedron it falls and what are its barycentric coordinates. Suppose it falls in a tetrahedron with vertices as r_k, r_l, r_m and r_n . Then the barycentric coordinates (b_1, b_2, b_3, b_4) of the point ρ is given by :

$$(b_1, b_2, b_3, b_4) = \left(\frac{V_k}{V}, \frac{V_l}{V}, \frac{V_m}{V}, \frac{V_n}{V} \right)$$

where V_k is the volume of tetrahedron (r_l, r_m, r_n, ρ) and so on and V is the volume of the tetrahedron (r_k, r_l, r_m, r_n) . Using these coordinates and the values of the displaced control points we obtain the new value ρ' of the point ρ as :

$$\rho' = b_1 * s_k + b_2 * s_l + b_3 * s_m + b_4 * s_n$$

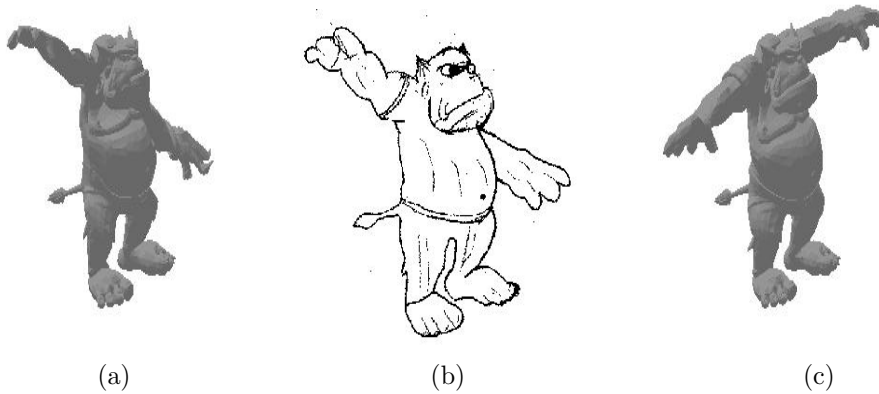


Figure 5: (a) The deformed model as seen from the computed P , (b) the reference drawing, and (c) the 3D base model as seen from the computed P .

where each s_i is the displaced control point of r_i .

Figure 5 (a) shows the deformed model as seen from the computed projection matrix. For comparison, the reference drawing (b) and the undeformed base model as seen from the same projection matrix (c) are also shown.

The accuracy of our approach depends on the accuracy of the corresponding points. But practically, as we need only a few corresponding points (typically as were in the running example) we can easily take fairly accurate corresponding points such as corner points or distinguishable points of the model (like the tips of ears, tail, feet, fingers etc. in the running example).

3. Application to View-Dependent Geometry

As was illustrated in the previous section our approach can be used to deform a 3D

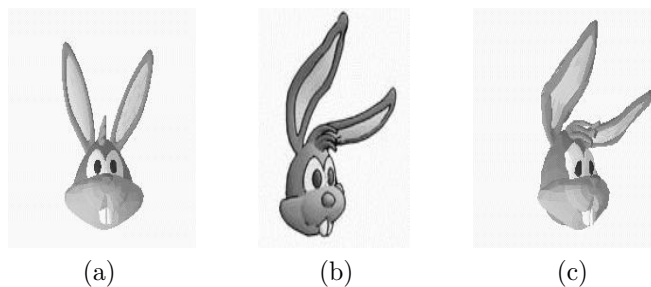


Figure 6: (a) The base model, (b) a reference drawing, and (c) the corresponding key deformation (c).

model according to a reference drawing such that the deformed object looks like the reference drawing from the intended view-point. This has got direct application in obtaining view-dependent models. Figure 6 shows a view-dependent model obtained using our approach. The model taken is the same as was used by Rademacher in [1]. Figure 6 (a) shows the base model, figure 6 (b) shows a reference drawing and figure 6 (c) shows the corresponding key deformation we obtained for the reference drawing.

After the view-dependent model is created out of the original 3D base model and few reference drawings, we can use the interpolation technique of [1] to generate view of the object from any arbitrary view-point such that the view is consistent with the views as specified for the key view-points. Figure 7 shows an animation sequence which interpolates the view of an undeformed base model to its key deformation. Figure 8 shows a similar animation sequence for the model used in the previous section. Both these animation sequences can be seen on the web page at <http://www.cse.iitd.ernet.in/~pkalra/avdg>.

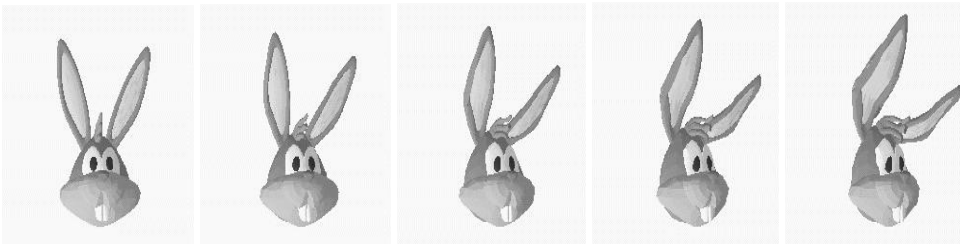


Figure 7: Animation sequence from one view of the base model to the intended view of the deformed model.



Figure 8: Animation sequence from one view of the base model to the intended view of the deformed model.

4. Conclusions

We have presented a semiautomatic approach to obtain view dependent deforma-

tions given the key views using projective camera model. We have shown results qualitatively similar to [1]. Except for manually specifying the corresponding points, rest of our approach is automated. Our approach relieves the user from manually manipulating the 3D mesh as in [1] which required artistic skill from the user. Instead the user has a simpler task of specifying a few corresponding points between two images: the reference drawing and a projection of the 3D model. The manual part in our approach requires no artistic skill from the user. Automatic feature detection can be employed to further automatize this process. Such results may find use in making view-dependent models and hence in making effective cartoon animations.

Acknowledgements

Authors would like to thank Paul Rademacher for making the Bunny's base model and the respective reference drawing available used in Section 3. The model has been downloaded from <http://www.people.fas.harvard.edu/~turian/vdge-extension/>. We also extend our thanks to Gaurav Agarwal and Dinesh Raathi for providing the data for the base model used in Section 2.

References

1. Paul Rademacher. View-dependent geometry. In *Proceedings of SIGGRAPH 99*, pages 439-446. August 1999. ACM.
2. Joseph Turian and Stephen Wollman. Extending View-Dependent Geometry. <http://www.people.fas.harvard.edu/~turian/vdg-extension/>.
3. Olivier Faugeras. Three-Dimensional Computer Vision: A Geometric Approach. MIT Press, Cambridge, Massachusetts, 1993.
4. Thomas Sederberg and Scott Parry. Free-Form Deformation of Solid Geometric Models. In *Proceedings of SIGGRAPH 86*, pages 151-160. New York, Aug 1986. ACM.
5. Karan Singh and Eugene Fiume. Wires: A Geometric Deformation Technique. In *Proceedings of SIGGRAPH 98*, pages 405-414. New York, July 1998. ACM.
6. Bechmann D. Space deformation models survey, *Computer and Graphics*, 18, 4, Pergamon, pages 571-586. 1995.
7. Farin G. *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic Press, Second Edition, 1990.

Photo and Bibliography



Rohit Jaivant Kate received his Bachelor of Technology degree in Computer Science and Engineering from Indian Institute of Technology, Delhi, India in 2000. He is pursuing Doctoral degree in Computer Science at the University of Texas at Austin, USA.



Prem Kalra is an associate professor in Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi. Before joining IIT Delhi, he was at MIRALab, University of Geneva (Switzerland). He obtained his PhD in computer science from Swiss Federal Institute of Technology, Lausanne in 1993. His research interests include 3D visualization, image based rendering and animation, human modeling and animation, and virtual reality.



Subhashis Banerjee did B. E. (Electrical Engineering) from Jadavpur University, Calcutta in 1982, and M. E. (Electrical Engineering) and Ph. D. from the Indian Institute of Science, Bangalore, in 1984 and 1989 respectively. Since 1989 he has been on the faculty of the Department of Computer Science and Engineering at IIT Delhi where he is currently a Professor. His research interests include Computer Vision and Real-time Embedded Systems.