View Morphing in the Movies - James Chen, Adam Smith

View Morphing in the Movies

James Chen, Adam Smith

Background

History

Hey, remember bullet time? When The Matrix came out in 1999, audiences were blown away by this amazing technical achievement. This effect subsequently made its way to other forms of media: other movies (Swordfish), TV shows (Smallville), video games (Max Payne) and even Superbowl XXV. Although trademarked as "bullet time" by Warner Brothers, the effect does not have to include bullets. It is simply the rotation of a camera view around a scene as it unfolds in slow motion time. This cannot be achieved in reality as the camera would have to rotate at an incredible speed, thus special effects are employed to achieve this effect (in The Matrix some scenes approached a frame rate of around 12,000 frames per second, each at a different angle around a target character). The main technique used is a type of image morphing known as "view morphing."

For The Matrix, the Wachowski Brothers would take some still pictures at various portions of the desired camera path. They would then use computers to map out a camera trajectory using the earlier still photos as key frames. Then, they would build a track, and using a large set of sophisticated still cameras they would film the scene [9]. Despite using many expensive cameras, this would still not be enough individual frames to create a convincing bullet time effect. This is where view morphing comes in. They would use view morphing software to interpolate between camera frames to allow the frame rate to become incredibly high. This technique also allowed the filmmakers to have complete control over the flow of the action. Certain actions of the animation could be sped up or slowed down, or a given view could be lingered upon for an extra beat for effect.

The concept of view morphing is actually an extension of the already existing image morphing technique. These techniques have been used in the special effects industry since the late 80's and 90's. Notable scenes include the 1988 film Willow in which morphing techniques pioneered by ILM were used to change animals to humans [1][2], and the 1991 Michael Jackson music video Black Or White where people of different races and genders dance and sing as they are morphed into one another [3]. Morphing first appeared in the 1986 film The Golden Child, but Willow was the first time it was used extensively. Morphing is regarded as commonplace in films with a moderate amount of special effects in modern cinema.

List of Media using Bullet Time

Movies

- Charlie's Angels
- The Matrix Franchise
- Shrek
- Swordfish
- Willow (Morphing video)

Video Games

- Conker's Bad Fur Day
- F.E.A.R.
- Max Payne (Example video)
- Metal Gear Solid: The Twin Snakes

Television

- Superbowl XXV
- Smallville
- Howard Stern intro

Music Videos

- Michael Jackson - Black or White (video)

- Korn - Freak on a Leash

A Technical Look at View Morphing

The idea behind image morphing is to interpolate between pixels of images to produce a smooth transition between them, however the resulting transitions do not always appear natural. One of the main causes for this unnatural transition is that morphing methods (and there are numerous ones) do not always account for changes in viewpoint or object pose [4]. View Morphing is a technique that employs these classical image morphing techniques but implements additional constraints so that the resulting final morph does not distort the object of interest in the images (known as a "shape preserving" morph). Because objects are not distorted in the morph, the resulting images show what one would see if one were to physically move the camera or object between the configurations in the images. If the two images used in the morph were taken of the same scene, view morphing would then produce the "bullet time" effect mentioned above. In effect, one would be able to interpolate a "virtual camera" anywhere in between the physical location of the two cameras that took the initial images, and capture what the virtual camera sees. Visually, this would be akin to freezing a scene in time, and moving the viewer along a path of physical cameras setup around the scene. However, there are drawbacks to the resulting images. View Morphing only operates on 2D images, and is in effect, trying to interpolate 3D transformations. Since only 2D data is used, a surface being morphed can only be accurately transitioned if it is visible in both interpolating images. On areas of the image where this is not the case, depending on the image morphing technique used, one could see visual artifacts such as ghosting and shape distortions. As such, this technique yields better results when the interpolating images taken of a scene have physically close optical centers.

Algorithm

Presented below is the algorithm for View Morphing as outlined by Seitz and Dyer et. al [4]

Shape Preservation

View Morphing works by encasing the classical image morphing technique around a pre and post warping of the images which ensure shape preservation. Generic Images I0 and I1 passed into an image morph are not generally shape preserving because they are not **parallel views**. Parallel views refers to the two images I1 and I2 having parallel image planes. Intuitively, this means interpolated views will also have an image plane that is parallel to I0 and I1. Mathematically this means the two projection matrices of I0 and I1 are:

$$\Pi_0 = \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\Pi_1 = \begin{bmatrix} f_1 & 0 & 0 & -f_1 C_x \\ 0 & f_1 & 0 & -f_1 C_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

which assumes that the optical center of I0 (call this C0) is at the origin with a focal length of f0, and the optical center of I1 (call this C1) is translated by Cx and Cy with a new focal length of f1.

Linear interpolation from these two to a point between C0 and C1 will yield a new matrix:

$$\Pi_s = (1-s)\Pi_0 + s\Pi_1 = \begin{bmatrix} (1-s)f_0 + sf_1 & 0 & 0 & -sf_1 C_x \\ 0 & (1-s)f_0 + sf1 & 0 & -sf_1 C_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where s represents how far along C0 and C1 to interpolate. The new optical center and focal length for this interpolated position is (s*Cx, s*Cy) and (1-s)*f0 + s*f1 respectively.

This is analogous to moving the camera along the line C0 to C1 and zooming continuously. Since the object remains the same in the interpolated parallel view, it is shape preserving. This result only depends on the equality of the third rows of the two interpolating projection matrices. There are other special cases of this general condition, e.g. if the last row is [0 0 0 1] for both projection matrices we have orthographic projections which are also shape preserving.

Pre Warp

So we know that in order to apply an image morph so that the object of interest in a scene is preserved, the image planes must be in parallel. It is generally not the case that two images taken are in parallel view, which is why the pre-warp step is needed in view morphing. This step turns two non parallel image planes into parallel image planes so they can be linearly interpolated and retain shape preservation.

The first step in calculating the Pre Warped images is to calculate the fundamental matrix of the two views. The fundamental matrix is a 3 x 3 rank 2 matrix that relates corresponding points in the two images [5] and is described by the condition:

$$x_0^T F x_1 = 0$$

where x0 and x1 are corresponding points.

The fundamental matrix of two corresponding images can be determined if at least eight corresponding points are known using Hartley's Algorithm.

Hartley's eight point algorithm is fairly straight forward. It uses the condition imposed for the definition of a fundamental matrix (shown above) to form a linear system that can be solved with 8 known points [6]. Assume we have two corresponding points a and b, and the fundamental matrix that relates them is F. By the definition we have:

$$a^T F b = 0$$

where

$$a = \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ 1 \end{bmatrix} \text{ and } F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Multiplying the matrix formula out yields:

$$a_1 b_1 f_{11} + a_1 b_2 f_{12} + a_1 f_{13} + a_2 b_1 f_{21} + a_2 b_2 f_{22} + a_2 f_{23} + b_1 f_{31} + b_2 f_{32} + f_{33} = 0$$

which can be re-written as:

$$c = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_1 \\ a_2 b_1 \\ a_2 b_2 \\ a_2 \\ b_1 \\ b_2 \\ 1 \end{bmatrix} \text{ and } f' = \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}$$

with the condition:

$$f \cdot c = 0$$

the vector form of the fundamental matrix is unchanged, but we can generate a c vector for every pair of corresponding points. If we generate eight linearly independent vectors of c (from eight corresponding points), we can solve for the 9-dimensional vector of the fundamental matrix.

Solving for f is straightforward in a program like matlab (which is what will be used). Eight vectors of c can be formed from the eight corresponding points and grouped into a matrix. The Singular Value Decomposition of this matrix is taken, and the right singular vector (V) corresponding to the eigenvalue of 0 (this will be the last column vector of V) is taken as the preliminary vector form of f (call this f_prelim). However we must also satisfy the internal constraint of the fundamental matrix (it is rank 2 so one of its singular values is 0). Thus an additional SVD is taken of f_prelim, and the singular matrix of f_prelim is modified so the last singular value is 0, and the matrix is rebuilt again to form the final fundamental matrix.

Once the fundamental matrix has been calculated, the two pre-warp transforms can be determined. The pre-warp transforms are projective transformation matrices that reprojects the image plane of an image I to another, I'. In our scenario, we want I' to be in the same parallel plane for both images. Once we know the fundamental matrix F, we can determine the two projective transforms H0 and H1. The method described by Seitz and Dyer et. al [4] applies a rotation to make the image planes parallel, followed by an affine transformation to align the scanlines.

The *epipoles* of each image are first needed to calculate the angles of rotation. These are the unit eigenvectors of F and F_transpose corresponding to eigenvalues of 0. The epipoles represent the projection of the optical center of the *other* camera. It is important to note that this only works if the epipoles are outside the field of view of the other camera (otherwise this is known as a *singular* view and is not possible). Choosing an arbitrary axis of rotation for each image d0, d1 (can use the eigenvectors), we can then form the rotational and translational matrices needed to form H0 and H1, shown mathematically below:

Given

$$F, F'$$

let

$$e_0 = [e_0^x, e_0^y, e_0^z], e_1 = [e_1^x, e_1^y, e_1^z]$$

be their eigenvectors, respectively.

Choose

$$d_0 = [-e_0^y, e_0^x, 0]^T, d_1 = [-Fe_0^x, -Fe_0^y, 0]$$

We can now calculate:

$$\theta_0 = -\frac{\pi}{2} - tan^{-1}(\frac{d_0^y e_0^x - d_0^x e_0^y}{e_0^z})$$
$$\theta_1 = -\frac{\pi}{2} - tan^{-1}(\frac{d_1^y e_1^x - d_1^x e_1^y}{e_1^z})$$

Knowing theta0 and theta1 lets us calculate the rotational matrices of these angles around d0 and d1 to make I0 and I1 parallel:

$$R_{\theta_0}^{d_0} = \begin{bmatrix} (d_0^x)^2 + (1-(d_0^x)^2)cos\theta_0 & d_0^x d_0^y(1-cos\theta_0) & d_0^y sin\theta_0 \\ d_0^x d_0^y(1-cos\theta_0) & (d_0^y)^2 + (1-(d_0^y)^2)cos\theta_0 & d_0^x sin\theta_0 \\ -d_0^y sin\theta_0 & d_0^x sin\theta_0 & cos\theta_0 \end{bmatrix}$$

$$R_{-\theta_0}^{d_0} = \begin{bmatrix} (d_0^x)^2 + (1-(d_0^x)^2)cos(-\theta_0) & d_0^x d_0^y(1-cos(-\theta_0)) & d_0^y sin(-\theta_0) \\ d_0^x d_0^y(1-cos(-\theta_0)) & (d_0^y)^2 + (1-(d_0^y)^2)cos(-\theta_0) & d_0^x sin(-\theta_0) \\ -d_0^y sin(-\theta_0) & d_0^x sin(-\theta_0) & cos(-\theta_0) \end{bmatrix}$$

$$R_{\theta_1}^{d_1} = \begin{bmatrix} (d_1^x)^2 + (1-(d_1^x)^2)cos\theta_1 & d_1^x d_1^y(1-cos\theta_1) & d_1^y sin\theta_1 \\ d_1^x d_1^y(1-cos\theta_1) & (d_1^y)^2 + (1-(d_1^y)^2)cos\theta_1 & d_1^x sin\theta_1 \\ -d_1^y sin\theta_1 & d_1^x sin\theta_1 & cos\theta_1 \end{bmatrix}$$

We also can also rotate the images so that their epipolar lines are horizontal (the images would share equivalent scanlines, simplifying the morphing step to a scanline interpolation). We first need to find the horizontal rotation angle phi:

let

$$[\tilde{e}_0^x, \tilde{e}_0^y, 0]^T = R_{\theta_0}^{d_0} e_0$$
$$[\tilde{e}_1^x, \tilde{e}_1^y, 0]^T = R_{\theta_1}^{d_1} e_1$$

be the new rotate epipoles. Then:

$$\phi_0 = -tan^{-1}(\frac{\tilde{e}_0^y}{\tilde{e}_0^x})$$
$$\phi_1 = -tan^{-1}(\frac{\tilde{e}_1^y}{\tilde{e}_1^x})$$

and the Rotation matrices for phi are:

$$R_{\phi_0} = \begin{bmatrix} cos\phi_0 & -sin\phi_0 & 0 \\ sin\phi_0 & cos\phi_0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$R_{\phi_1} = \begin{bmatrix} cos\phi_1 & -sin\phi_1 & 0 \\ sin\phi_1 & cos\phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The new Fundamental Matrix of the rotated image planes is:

$$\tilde{F} = R_{\phi_1} R_{\theta_1}^{d_1} F R_{-\theta_0}^{d_0} R_{-\phi_0}$$

which has the form

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & b & c \end{bmatrix}$$

The second image is translated and scaled by:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -a & -c \\ 0 & 0 & b \end{bmatrix}$$

So the final pre-warp transform matrices are:

$$H_0 = R_{\phi_0} R_{\theta_0}^{d_0}$$
$$H_1 = T R_{\phi_1} R_{\theta_1}^{d_1}$$

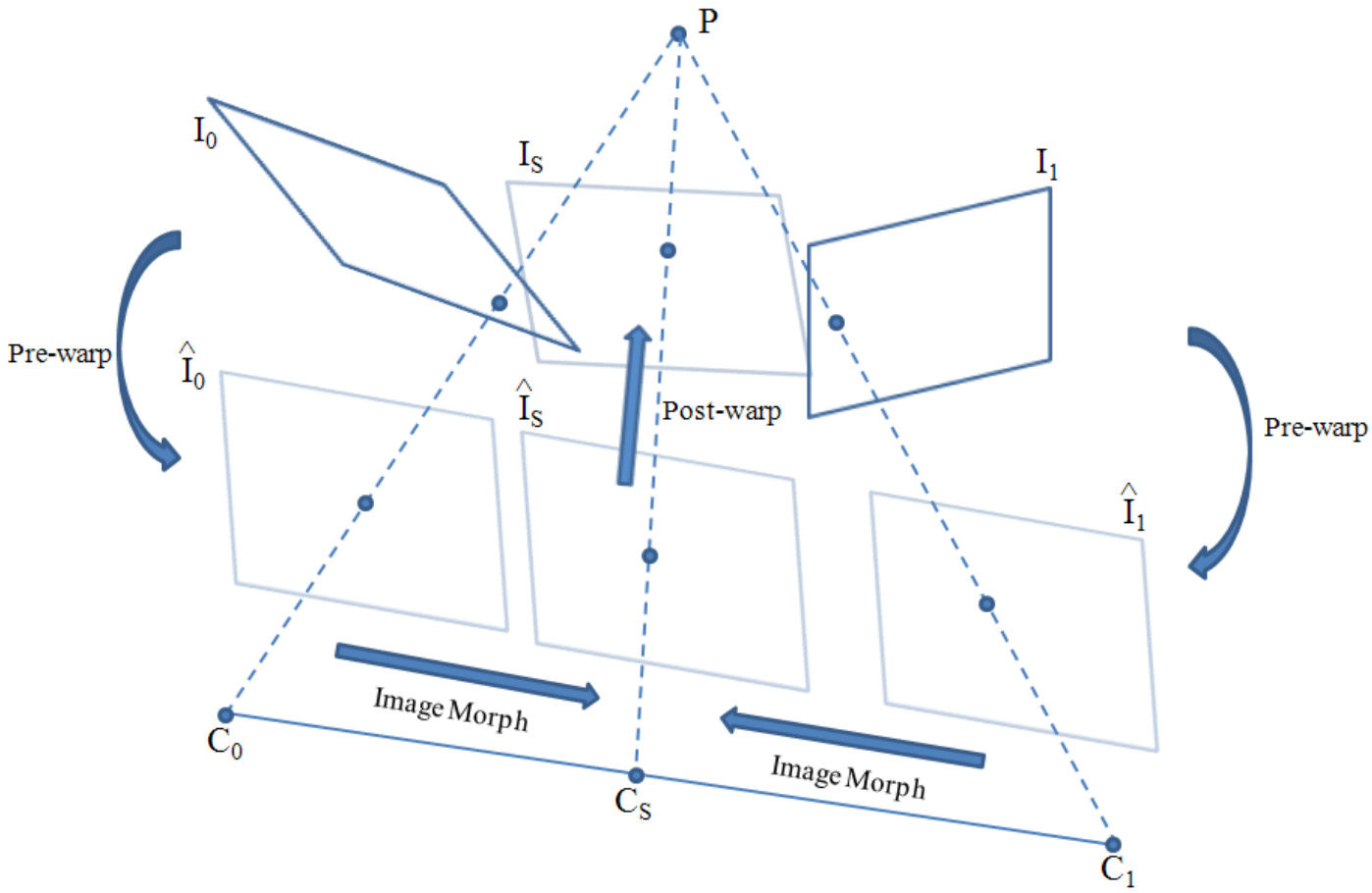
Image Morph

The image morphing step can be done in numerous ways. Seitz and Dyer suggest feature based image morphing which was developed by Beier and Neely et. al [8] since it is one of the more popular image morphing algorithms in use. This method requires the user to input corresponding lines on the two morphing images, and weights nearby points by their proximity to the corresponding lines (this is done for each line). The weighted average displacements is then used to compute the final destination point from the source point. It is a fairly straightforward algorithm, and more information can be read in the paper [8]. However, this portion is not the focus of the View Morphing algorithm, and is in fact, interchangeable for any type of linear morphing algorithm, thus we will not go into it in depth.

Post Warp

The Post Warping is needed to bring the warped interpolated image to its unwarped state corresponding to the two original images passed in to be view morphed. Seitz and Dyer et. al claim it is easier to provide the postwarp transform Hs indirectly by establishing constraints on inbetween images than to explicitly specify the 3x3 matrix. This can be done by specifying four control points in the prewarped images and original images. Knowing the correspondences of four common points in the prewarped and original images implicitly gives us a way to find a post warp at any point along the view morph. Since pre-warped images are parallel views, a simple linear interpolation of the four points specified in these images will give us the prewarped version of our final image. Knowing the post warping from the original correspondence, we can thus apply the same post warp at this interpolated position to get a post warped final view morphed image.

The image below depicts graphically the view morphing procedure across its three steps:



I0 and I1 are the original, unparalleled images. I0_hat and I1_hat are the pre-warped versions of these images with optical centers C0 and C1, respectively. They are linearly interpolated (image morphed) to an inbetween point s, yielding image Is_hat with optical center Cs, and then post warped into the view morphed image Is.

Example of View Morphing

The algorithms and code outlined by Seitz and Dyer et. al are very straightforward and easy to implement in matlab. As such, a group on the internet has already done so: http://www.duke.edu/~kmg/cps296/code/

The code base is implemented exactly as Seitz and Dyer described and even uses Beier and Neely's method as the image morphing algorithm. We used this code on our own images to generate the image morphs shown below. Minor changes were done to fix a bug we came across that crashed matlab due to inconsistent prewarp image sizes, as well as changes to morphing parameters to allow for more interpolated frames and input features.

Modified Code

The modified code used to generate the images shown below: view_morph_code.zip

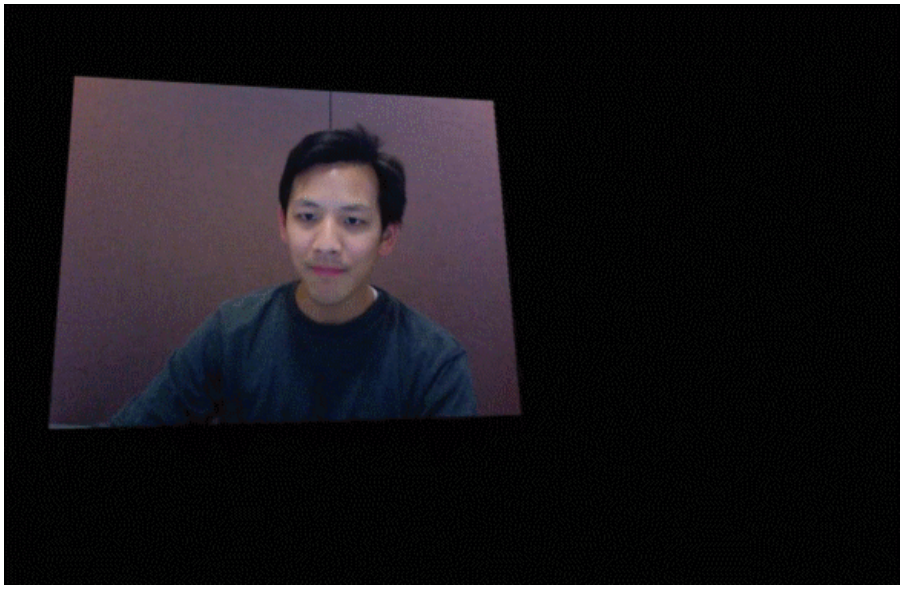
The first saved workspace containing all the variables after the initial fundamental matrix generation (line 58 of ViewMorph.m) of the closer camera setup near_workspace.mat

The first saved workspace containing all the variables after the initial fundamental matrix generation (line 58 of ViewMorph.m) of the farther camera setup far_workspace.mat

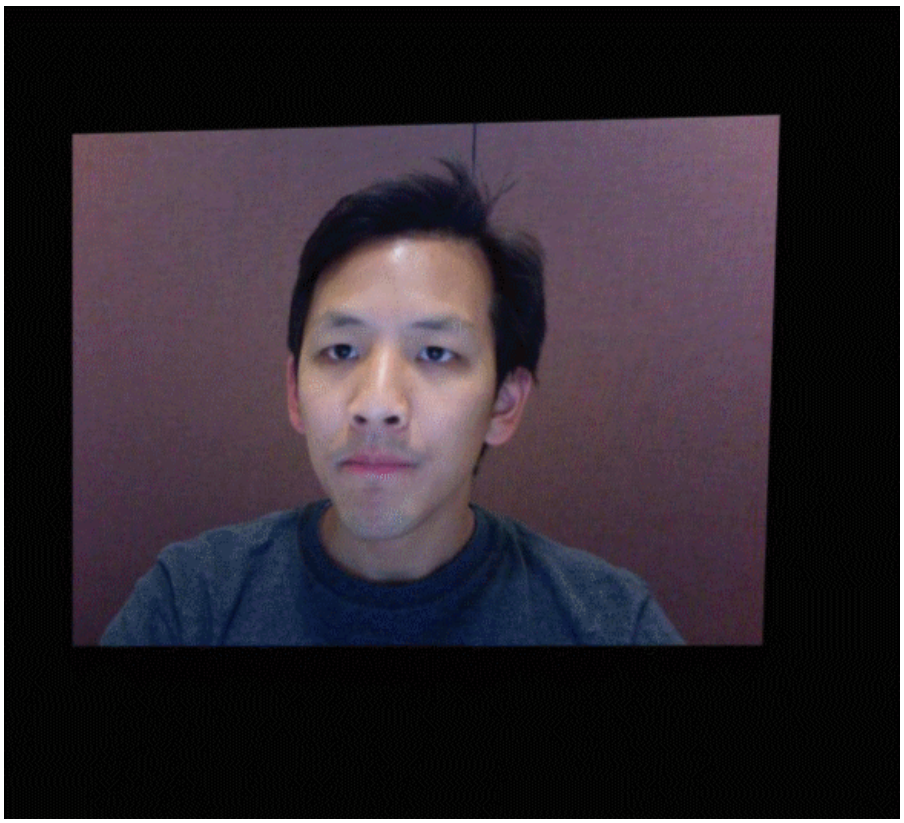
The workspaces of the setups were posted as the setup code can be inconsistent. If the points sent into Hartley's algorithm do not have strong linear independence, the pre-warping can become quite unpredictable. The two posted above have very good linear independence and corresponding pre-warps. One simply has to run the code from line 58 and on in ViewMorph.m (inputting the four corresponding post warp points, and inputting Feature Morph lines)

Results

The first setup was done with the interactive webcam on an iMac computer. The subject is situated approximately 2 feet away from the camera, and the distance between the two cameras is approximately 10 inches which gives an angular difference of roughly 24 degrees. Because this angular difference is so large, we see a lot of artifacts that crop up in the image morph. Lots of ghosting effects arrise on the edges of the face since these areas are more likely not not be seen in both images. The image warping also becomes a bit unstable in these regions. 50 view morphed frames were interpolated in between.



The second setup was done similarly except the distance between the two cameras is shortened to 4 inches. This yields an angular difference of about 9 degrees and a more suitable morph with less discrepancies in the interpolated images. Ghosting still arises on the edges of the face, but the facial expressions in between (since the features morph lines were concentrated on facial features), are more consistent and resistant to any distortion. It's interesting to note that at the midway morph point, the face has the effect of looking directly at the user, however, neither of the two images used to make this morph were done with the subject looking into the camera. 50 view morph frames were interpolated in between.



Limitations/Observations

The above results show that more artifacts arise as the angle between the subject and its two physical cameras increase. These artifacts can be alleviated through post processing such as blending operations, however, the best way to get quality images out is to put quality images in. In terms of view morphing, this means minimizing this angular difference as much as possible. The view morphs shown above were done with relative close proximity of the subject to the cameras. This means more details in the shot, as well as a larger angular difference. In the setup used in The Matrix [9], we can see the cameras used are stationed much farther from the subject. This yields a smaller angular difference, as well as less resolution in the subject which make ghosting artifacts much less observable and easier to fix through post processing.

References

[1] http://en.wikipedia.org/wiki/Morphing

[2] http://www.youtube.com/watch?v=uLUyuWo3pG0

[3] http://www.youtube.com/watch?v=F2AitTPI5U0

[4] Seitz SM, Dyer CR, "View Morphing", Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pgs 21-30, 1996

[5] http://en.wikipedia.org/wiki/Fundamental_matrix_(computer_vision)

[6] http://en.wikipedia.org/wiki/Eight-point_algorithm

[7] http://www.filmscouts.com/scripts/matinee.cfm?Film=matrix&File=bul-tim

[8] Beier, T. & Neely, S. (1992) Feature-based image metamorphosis, Computer Graphics, 26(2), 35-42

[9] http://www.instructables.com/id/Matrix-Bullet-Time-the-REAL-way/step2/Technology/

2011-03-29 17:45