Image/View Morphing and Warping

CS334

Daniel G. Aliaga
Department of Computer Science
Purdue University
Motivation – Rendering from Images

- Given
  - left image
  - right image
- Create intermediate images
  - simulates camera movement
Related Work

• Panoramas (e.g., QuicktimeVR, etc)
  – user can look in any direction at few given locations but camera translations are not allowed...
Topics

• Image morphing (2D)
• View morphing (2D+)
• Image warping (3D)
Topics

• Image morphing (2D)
• View morphing (2D+)
• Image warping (3D)
Image Morphing
Image Morphing

• Identify correspondences between input/output image

• Produce a sequence of images that allow a smooth transition from the input image to the output image
Image Morphing

1. Correspondences
Image Morphing

1. Correspondences
Image Morphing

1. Correspondences
Image Morphing

1. Correspondences
Image Morphing

1. Correspondences
2. Linear interpolation

\[ P_k = (1 - \frac{k}{n}) P_0 + \frac{k}{n} P_n \]
Image Morphing
Image Morphing

Image morphing is not shape preserving
Topics

- Image morphing (2D)
- View morphing (2D+)
- Image warping (3D)
View Morphing
View Morphing

• Shape preserving morph
• Three step algorithm
  – Prewarp first and last images to parallel views
  – Image morph between prewarped images
  – Postwarp to interpolated view
Step 1: prewarp to parallel views

- Parallel views
  - same image plane
  - image plane parallel to segment connecting the two centers of projection

- Prewarp
  - compute parallel views $I_{0p}$, $I_{np}$
  - rotate $I_0$ and $I_n$ to parallel views
  - prewarp correspondence is $(P_0, P_n) \rightarrow (P_{op}, P_{np})$
Step 2: morph parallel images

- Shape preserving
- Use prewarped correspondences
- Interpolate $C_k$ from $C_0$ to $C_n$
Step 3: postwarp image

- Postwarp morphed image
  - create intermediate view
    - $C_k$ is known
    - interpolate view direction and tilt
  - rotate morphed image to intermediate view
View morphing
View morphing

• View morphing is shape preserving
View Morphing Examples

- Using computer vision/stereo reconstruction techniques
• Intuitively, how do you compute the matrix $M$ by which to transform $P_0$ to $P_{0p}$?
Image Transformations

• A geometric relationship between input \((u,v)\) and output pixels \((x,y)\)

  – Forward mapping:
    \[
    (x,y) = (X(u,v), Y(u,v))
    \]

  – Inverse mapping:
    \[
    (u,v) = (U(x,y), V(x,y))
    \]
Image Transformations

• General matrix form is

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

and operates in the “homogeneous coordinate system”.
Affine Transformations

• Matrix form is

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

and accommodates translations, rotations, scale, and shear.

• How many unknowns? How to create matrix?
Affine Transformations

- Transformation can be inferred from correspondences; e.g.,

\[
\begin{bmatrix}
u_i \\
v_i \\
w_i
\end{bmatrix} \leftrightarrow \begin{bmatrix}
x_i \\
y_i \\
z_i
\end{bmatrix}
\]

- Given \( \geq 3 \) correspondences can solve for T
Perspective/Projective Transformations

• Matrix form is

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & 1
\end{bmatrix}
\begin{bmatrix}
    u \\
    v \\
    w
\end{bmatrix}
= 
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
\]

and it accommodates foreshortening of distant line and convergence of lines to a vanishing point; also, straight lines are maintained but not their mutual angular relationships, and only parallel lines parallel to the projection plane remain parallel.
Perspective/Projective Transformations

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & 1
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix}
= 
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix}
\]

• How many unknowns?
• How many correspondences are needed?
Perspective/Projective Transformations

- Solve

\[
\begin{pmatrix}
  u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\
  u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\
  u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\
  u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\
  0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\
  0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\
  0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\
  0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3
\end{pmatrix} = \begin{pmatrix}
  A \\
  b
\end{pmatrix}
\]

where \( A \) is the vector of unknown coefficients \( a_{ij} \)
Topics

- Image morphing (2D)
- View morphing (2D+)
- Image warping (3D)
3D Image Warping

- **Goal:** “warp” the pixels of the image so that they appear in the correct place for a new viewpoint

- **Advantage:**
  - Don’t need a geometric model of the object/environment
  - Can be done in time proportional to screen size and (mostly) independent of object/environment complexity

- **Disadvantage:**
  - Limited resolution
  - Excessive warping reveals several visual artifacts (see examples)
3D Image Warping Equations

Some pictures courtesy of SIGGRAPH '99 course notes
(Leonard McMillan)
3D Image Warping Equations

\[ \dot{\mathbf{C}}_2 + t_2 P_2 \mathbf{x}_2 = \dot{\mathbf{C}}_1 + t_1 P_1 \mathbf{x}_1 \]

\[ t_2 P_2 \mathbf{x}_2 = \dot{\mathbf{C}}_1 - \dot{\mathbf{C}}_2 + t_1 P_1 \mathbf{x}_1 \]

\[ t_2 \mathbf{x}_2 = P_2^{-1}(\dot{\mathbf{C}}_1 - \dot{\mathbf{C}}_2) + t_1 P_2^{-1} P_1 \mathbf{x}_1 \]

\[ \frac{t_2}{t_1} \mathbf{x}_2 = \frac{1}{t_1} P_2^{-1}(\dot{\mathbf{C}}_1 - \dot{\mathbf{C}}_2) + P_2^{-1} P_1 \mathbf{x}_1 \]

\[ \mathbf{x}_2 = \underbrace{\frac{1}{t_1} P_2^{-1}(\dot{\mathbf{C}}_1 - \dot{\mathbf{C}}_2)}_{\delta} + \underbrace{P_2^{-1} P_1 \mathbf{x}_1}_{\mathbf{H}_{21}} \]
3D Image Warping Equations

McMillan & Bishop Warping Equation:
\[ x_2 = \delta(x_1) P_2^{-1} (c_1 - c_2) + P_2^{-1} P_1 x_1 \]

- Per-pixel distance values are used to warp pixels to their correct location for the current eye position.
3D Image Warping Equations

• Images enhanced with per-pixel depth
  [McMillan95]
3D Image Warping Equations

\[ P = C_1 + (c_1 + u_1 a_1 + v_1 b_1)w_1 \]

\[ w_1 = \frac{C_1 P}{C_1 P_1} \]

- \(1/w_1\) also called generalized disparity
- another notation \(\delta(u_1, v_1)\)
3D Image Warping Equations

\[ P = C_1 + (c_1 + u_1 a_1 + v_1 b_1) w_1 \]

\[ P = C_2 + (c_2 + u_2 a_2 + v_2 b_2) w_2 \]
3D Image Warping Equations

\[ u_2 = \frac{w_{11} + w_{12} \cdot u_1 + w_{13} \cdot v_1 + w_{14} \cdot \delta(u_1, v_1)}{w_{31} + w_{32} \cdot u_1 + w_{33} \cdot v_1 + w_{34} \cdot \delta(u_1, v_1)} \]

\[ v_2 = \frac{w_{21} + w_{22} \cdot u_1 + w_{23} \cdot v_1 + w_{24} \cdot \delta(u_1, v_1)}{w_{31} + w_{32} \cdot u_1 + w_{33} \cdot v_1 + w_{34} \cdot \delta(u_1, v_1)} \]
3D Image Warping Example
3D Image Warping Example

- DeltaSphere
  - Lars Nyland et al.
3D Image
Warping Example
3D Image
Warping Example
3D Image
Warping Example
3D Image
Warping Example
Disocclusions

• Disocclusions (or exposure events) occur when unsampled surfaces become visible...

What can we do?
Disocclusions

• Bilinear patches: fill in the areas

What else?
Rendering Order

The warping equation determines where points go...

... but that is not sufficient
Occlusion Compatible Rendering Order

- Remember epipolar geometry?
- Project the new viewpoint onto the original image and divide the image into 1, 2 or 4 “sheets”
Occlusion Compatible Rendering Order

- A raster scan of each sheet produces a back-to-front ordering of warped pixels
Splatting

• One pixel in the source image does not necessarily project to one pixel in the destination image
  – e.g., if you are walking towards something, the sample might get larger...

• A solution: estimate shape and size of footprint of warped samples
  – expensive to do accurately
  – square/rectangular approximations can be done quickly (3x3 or 5x5 splats)
  – occlusion-compatible rendering will take care of oversized splats
  – *BUT large splats can make the image seem blocky/low-res*
More Examples Using the DeltaSphere

- Lars Nyland *et al.*

courtesy 3rd Tech Inc.
spherical range panoramas

planar re-projection

Courtesy 3rd Tech Inc.
Jeep – one scan

Courtesy 3rd Tech Inc.
Complete Jeep model

Courtesy 3rd Tech Inc.
3D Modeling Can be Murder

3rdTech
Tour Into the Picture

Slides based on Horry et al. in SIGGRAPH
What is TIP?

Input: a single picture
What is TIP?

Output: a 3D-like animation
Videos

- Beach
- Interior
- Garden
- Lawson (inside)
- Lawson (outside)
Why TIP?

• Reconstruct detailed 3D model from the picture?
  – Very difficult, even by manually calibration
  – One picture is not sufficiently informative

• TIP supplies a much simplified 3D model (Box).
Process Flow

Figure 1. Process flow diagram
Process Flow

• (a) Get the input image
• (b) Define the background and extract the foreground mask.
• (c) Inpaint the background to make it complete.
  – We can use several image inpainting techniques today.
Process Flow

• (d) Build the spidery mesh by user input.
• (e) Modeling the 3D background.
The Method in the Original Paper

• In the original paper, the authors didn’t really reconstruct the 3D box model.
The Method in the Original Paper

- They provided an interactive controlling of a spidery mesh.

(a) Deformation of the inner rectangle  (b) Translation of the inner rectangle
The Method in the Original Paper

- Generate the new background

(a) Specified spidery mesh

(b) Deduced 2D polygon

(c) Estimating the vertices of the 3D rectangles
Alternative Method

• If we know the vanishing point and the assumed 3D structure, we can estimate the camera.

• Refer to § 8.9 Single view reconstruction, Multiple View Geometry in Computer Vision.

Fig. 8.26. Single view reconstruction. (a) Original image of the Fellows quad, Merton College, Oxford. (b) (c) Views of the 3D model created from the single image. The vanishing line of the roof planes is computed from the repetition of the texture pattern.
Process Flow

• Attach foreground object to the model
Process Flow

- Final Result

(f) Foreground model  (g) Camera positioning  (h) Rendered image
Images: Super cool!

- [Video]