Image Processing:

View and Image Morphing

CS334

Motivation - Rendering from Images

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

Overview

- Image morphing
- View morphing
  - image pre-warping
  - image morphing
  - image post-warping

Image Morphing Examples

Related Work

- Panoramas ([Chen95/QuicktimeVR], etc)
  - user can look in any direction at few given locations but camera translations are not allowed...

Quicktime VR Demo
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

1. Correspondences

\[ P_0, P_k, P_n \]

\[ \hat{P}_k = (1 - \frac{k}{n}) P_0 + \frac{k}{n} P_n \]
**Image Morphing**

- Image morphing is not shape preserving

**Overview**

- Image morphing
- View morphing
  - Image pre-warping
  - Image morphing
  - Image post-warping

**View Morphing Examples**

- 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
- Prewarp
  - Compute parallel views $I_{0p}$, $I_{np}$
  - Rotate $I_0$ and $I_n$ to parallel views
  - Prewarp correspondence is $(P_0, P_n) \rightarrow (P_{0p}, P_{np})$
Step 2: morph parallel images
- Shape preserving
- Use prewarped correspondences
- Interpolate $C_k$ from $C_0$ to $C_n$

Step 3: Postwarping
- 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 is shape preserving

View Morphing More Examples
- Using computer vision/stereo reconstruction techniques

Overview
- Image morphing
- View morphing, more details for synthetic rendering
  - image pre-warping
  - image morphing
  - image post-warping
Step 1: prewarp to parallel views

- Parallel views
  - use $C_0C_n$ for $x$ (ap vector)
  - use $(a_0 \times b_0) \times (a_n \times b_n)$ as $y$ (-bp)

- use $z = x \times y$
- use same pixel size
- use wider field of view

- prewarping using reprojection of rays
Step 1: prewarp to parallel views

- prewarping using reprojection of rays
  - look up all the rays of the prewarped view in the original view
- alternative: prewarping using texture mapping
  - create polygon for image plane
  - consider it texture mapped with the image itself
  - render “scene” from prewarped view
  - if you go this path you will have to implement clipping with prewarped plane
  - note: texture mapping in hardware

Step 1: prewarp to parallel views

- prewarping correspondences
  - for all pairs of correspondence $P_0, P_n$
    - project $P_0$ on $I_0^p$, computing $P_{0p}$
    - project $P_n$ on $I_n^p$, computing $P_{np}$
    - prewarped correspondence is $P_{0p} P_{np}$

Step 2: morph parallel images

- Image morphing
  - use prewarped correspondences to compute a correspondence for all pixels in $I_{0p}$
  - linearly interpolate $I_{0p}$ to intermediate positions
Step 2: morph parallel images

- Image morphing
  - use prewarped correspondences to compute a correspondence for all pixels in I₀ₚ
  - linearly interpolate I₀ₚ to intermediate positions
  - useful observation
    - corresponding pixels are on same line in prewarped views

- Preventing holes
  - use larger footprint (e.g., 2x2)
  - or linearly interpolate between consecutive samples
  - or postprocess morphed image looking for background pixels and replacing them with neighboring values

- Visibility artifacts
  - collision of samples

- Zbuffer of disparity
Step 2: morph parallel images

- Image morphing
  - use prewarped correspondences to compute a correspondence for all pixels in $I_0^p$
  - linearly interpolate $I_0^p$ to intermediate positions
  - useful observation
    - corresponding pixels are on same line in prewarped views
  - preventing holes
    - use larger footprint (e.g., 2x2)
    - or linearly interpolate between consecutive samples
    - or postprocess morphed image looking for background pixels and replacing them with neighboring values
  - visibility artifacts
    - collision of samples
    - Z-buffer of disparity
    - holes
    - morph $I_0^p$ to $I_k^p$
    - use additional views

Step 3: Postwarping

- create intermediate view
  - $C_k$ is known
  - current view direction is a linear interpolation of the start and end view directions
  - current up vector is a linear interpolation of the start and end up vectors
- rotate morphed image to intermediate view
  - same as prewarping