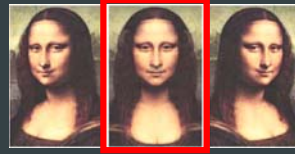


Capturing, Modeling, Rendering 3D Structures

View and Image Morphing

Motivation - Rendering from Images



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

[Seitz96]

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

Overview

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

Overview

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

Image Morphing Examples

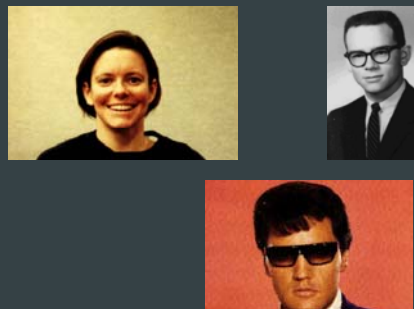


Image Morphing

- 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



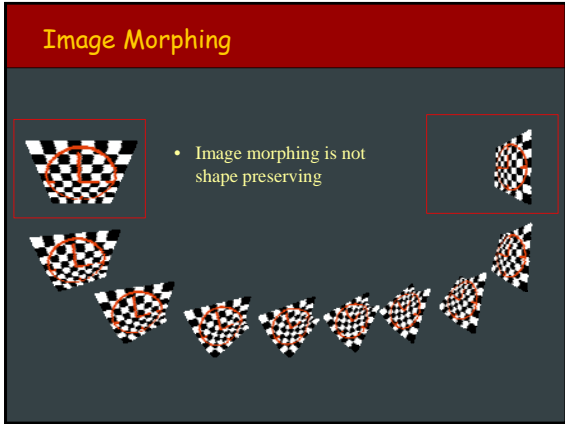
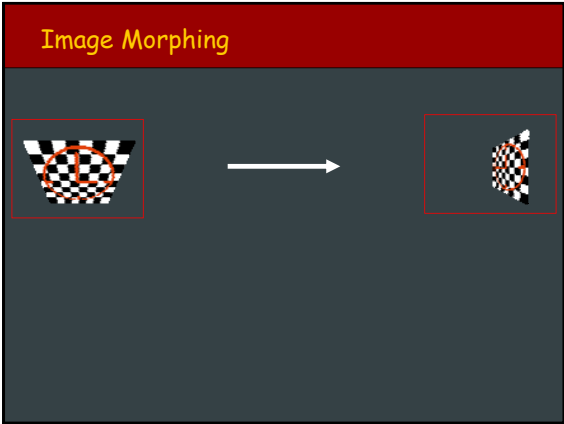
Image Morphing

1. Correspondences
2. Linear interpolation



$$\dot{P}_k = \left(1 - \frac{k}{n}\right) \dot{P}_0 + \frac{k}{n} \dot{P}_n$$

frame 0 frame k frame n



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



- ### 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

The diagram shows two cameras, C_0 and C_n , with centers of projection C_0 and C_n respectively. They are projecting onto an image plane P . The image planes are I_0 and I_n . The centers of projection are P_0 and P_n . The diagram shows how the image planes are rotated to be parallel to the segment connecting the two centers of projection.

- 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 $(P_0, P_n) \rightarrow (P_{0p}, P_{np})$

Step 2: morph parallel images

- Shape preserving
- Use prewarped correspondences
- Interpolate C_k from C_0 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

- 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

Step 1: prewarp to parallel views

- Parallel views
 - use C_0C_n for x (a_p vector)

Step 1: prewarp to parallel views

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

Step 1: prewarp to parallel views

- Parallel views
 - use C_0C_n for x (a_p vector)
 - use $(a_0 \times b_0) \times (a_n \times b_n)$ as y ($-b_p$)
 - use $z = x$ cross y

Step 1: prewarp to parallel views

- Parallel views
 - use C_0C_n for x (a_p vector)
 - use $(a_0 \times b_0) \times (a_n \times b_n)$ as y ($-b_p$)
 - use $z = x$ cross y
 - use same pixel size
 - use wider field of view

Step 1: prewarp to parallel views

- 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

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

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 $P_0 P_n$

Step 1: prewarp to parallel views

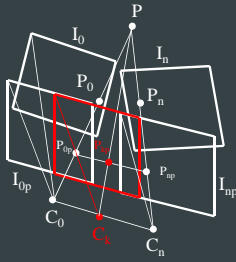
- prewarping correspondences
 - for all pairs of correspondence $P_0 P_n$
 - project P_0 on I_{op} , computing P_{0op}
 - project P_n on I_{op} , computing P_{nop}
 - prewarped correspondence is $P_{0op} P_{nop}$

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

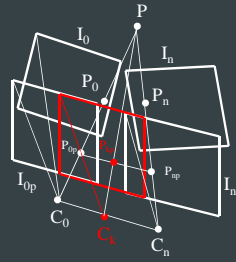
Step 2: morph parallel images

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



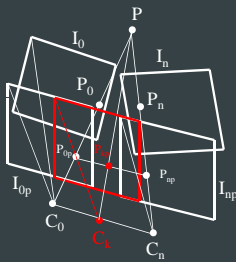
Step 2: morph parallel images

- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{op}
 - linearly interpolate I_{op} to intermediate positions
 - useful observation
 - corresponding pixels are on same line in prewarped views
 - preventing holes



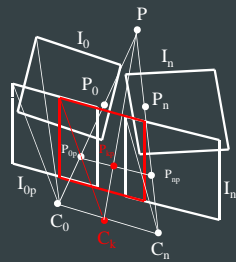
Step 2: morph parallel images

- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{op}
 - linearly interpolate I_{op} 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



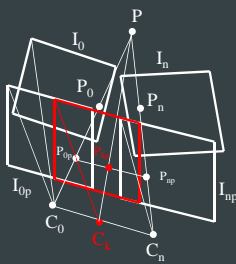
Step 2: morph parallel images

- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{op}
 - linearly interpolate I_{op} 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



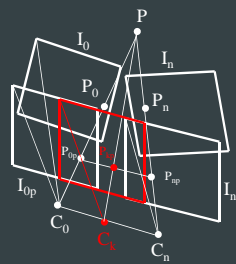
Step 2: morph parallel images

- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{op}
 - linearly interpolate I_{op} 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

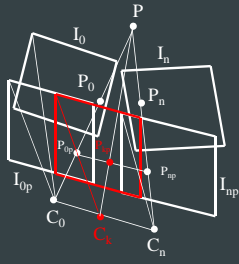


Step 2: morph parallel images

- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{op}
 - linearly interpolate I_{op} 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
 - holes

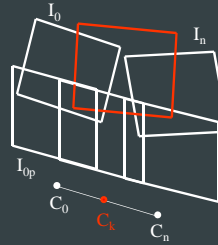


Step 2: morph parallel images



- Image morphing
 - use prewarped correspondences to compute a correspondence for all pixels in I_{0p}
 - linearly interpolate I_{0p} to intermediate positions
 - useful observation
 - corresponding pixels are on same line in prewarped views
 - preventing holes
 - use larger footprint (e.g., 2×2)
 - 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
 - holes
 - morph I_{0p} to I_{np}
 - 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