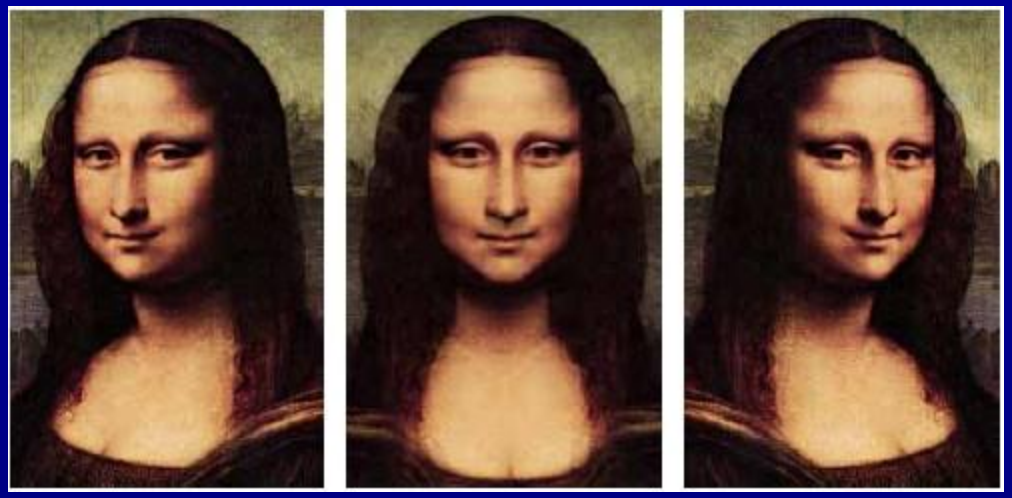
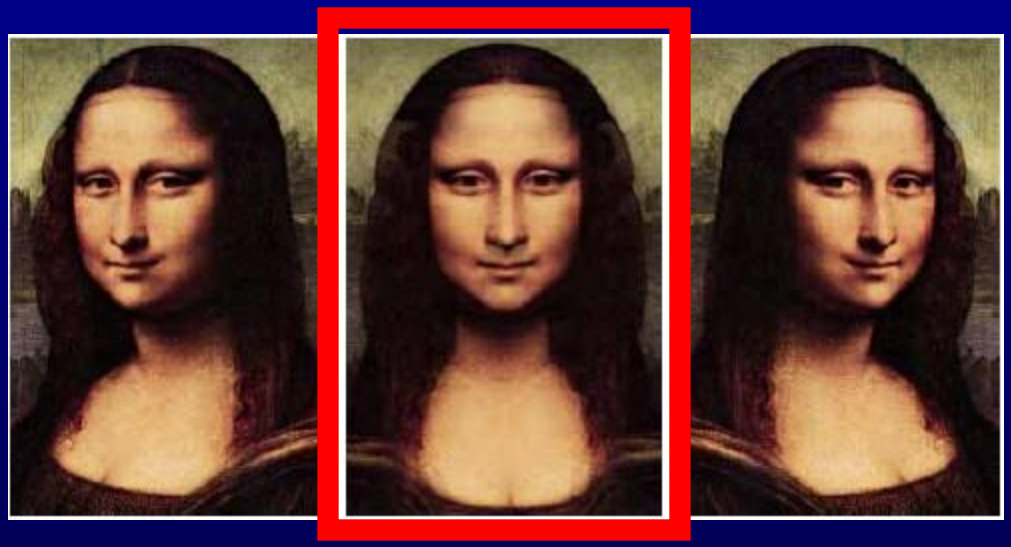


View morphing

Motivation – rendering from images



[Seitz96]

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

Previous work

- Panoramas ([Chen95], etc)
  - user can look in any direction at few given locations
- Image-morphing ([Wolberg90], [Beier92], etc)
  - linearly interpolated intermediate positions of features
  - input: two images and correspondences
  - output: metamorphosis of one image into other as sequence of intermediate images

Previous work limitations

- Panoramas ([Chen95], etc.)
  - no camera translations allowed
- Image morphing ([Wolberg90], [Beier92], etc.)
  - not *shape-preserving*
  - image morphing is also a morph of the object
  - to simulate rendering with morphing, the object should be rigid when camera moves

Overview

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

Overview

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

Image morphing

1. Correspondences

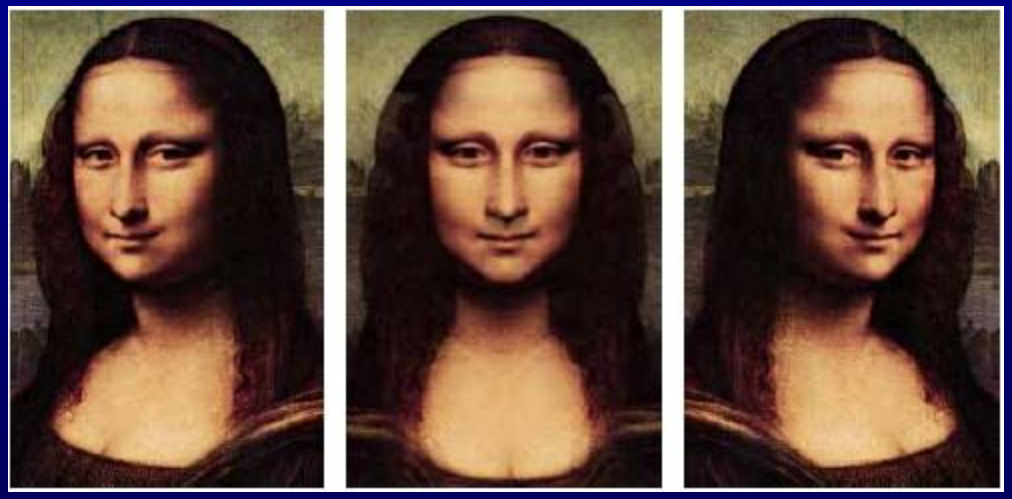


Image morphing

1. Correspondences

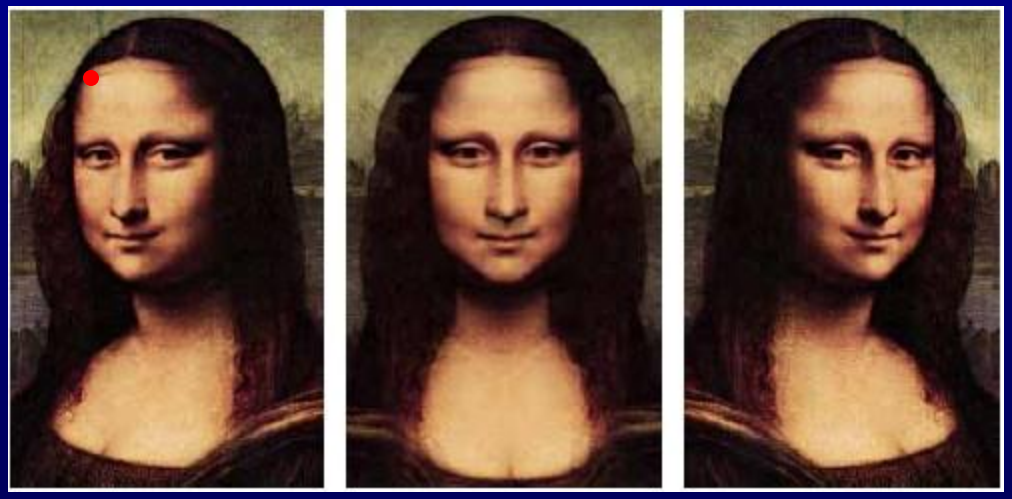


Image morphing

1. Correspondences

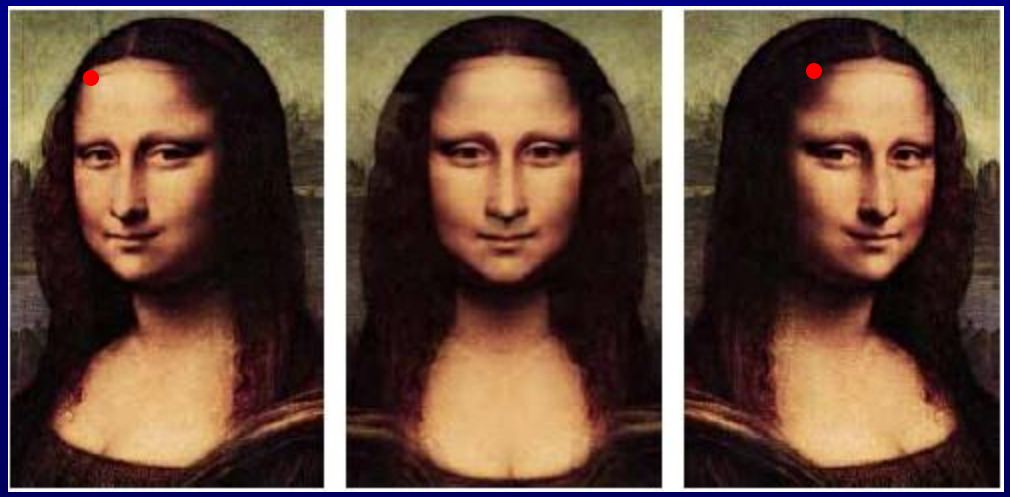


Image morphing

1. Correspondences

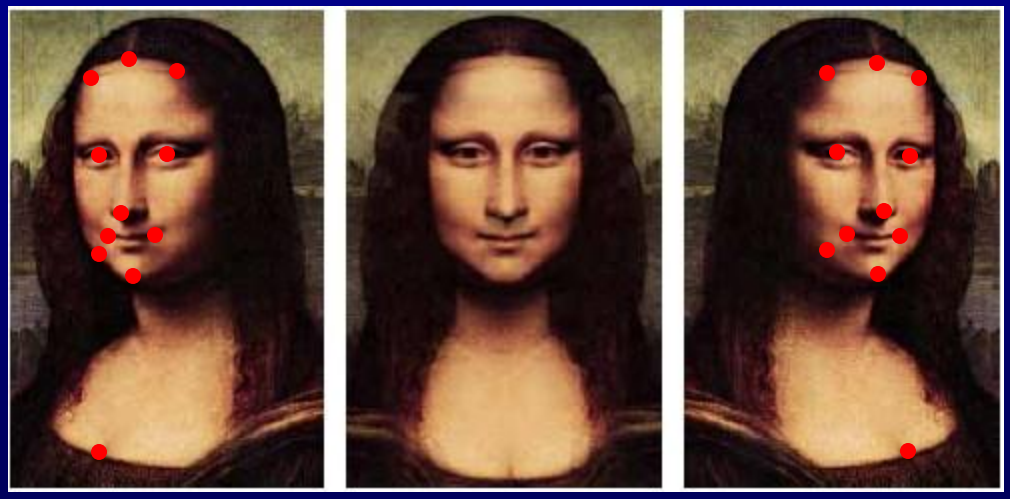
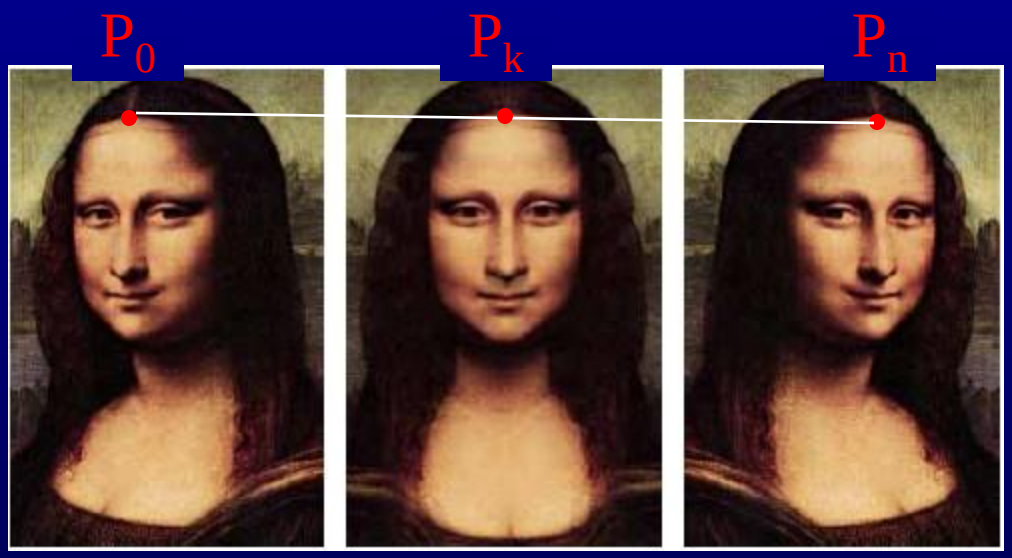


Image morphing



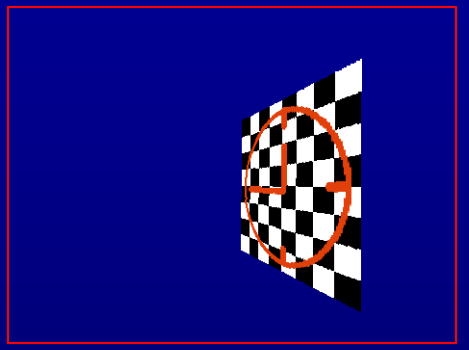
frame 0 frame k frame n

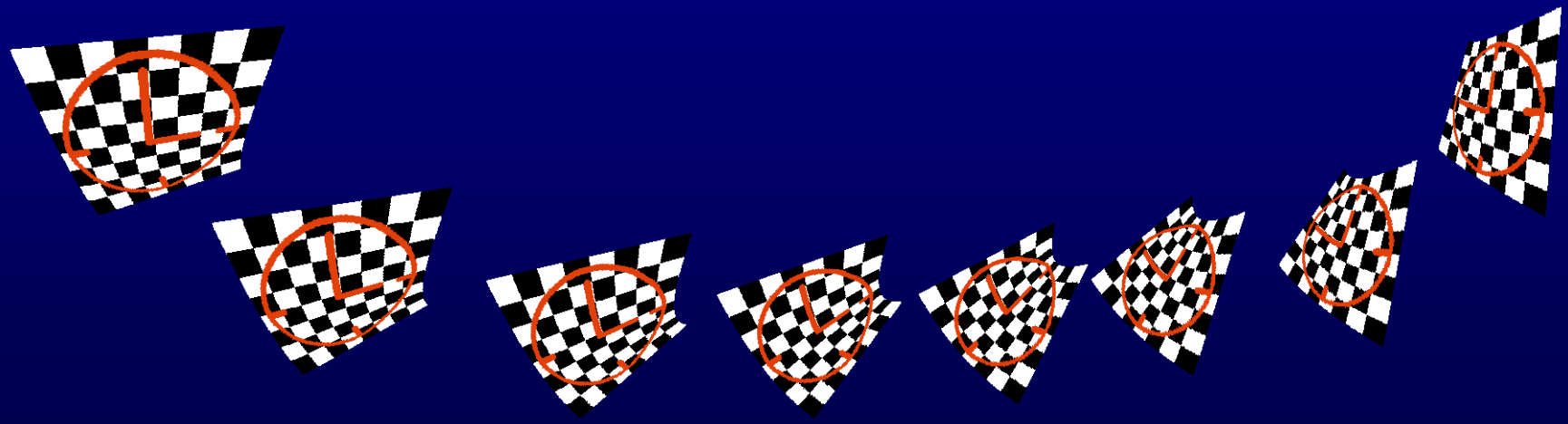
1. Correspondences
2. Linear interpolation

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

Image morphing

- Image morphing
  - not shape preserving





Early IBR research

Soft watch at moment of first explosion – Salvador Dali 1954

Overview

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

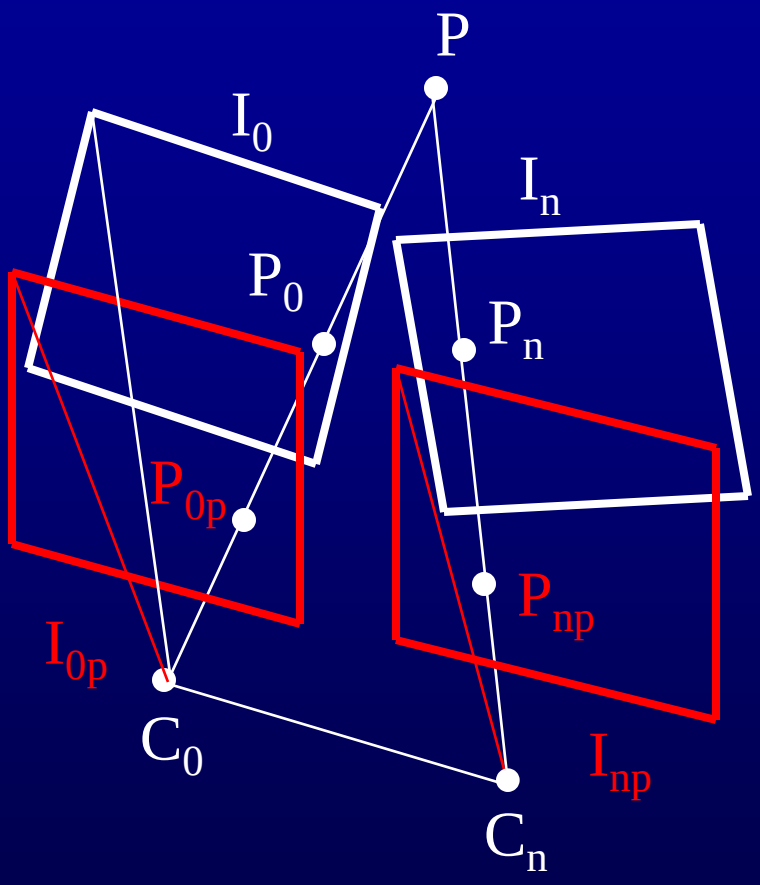
Overview

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

View morphing

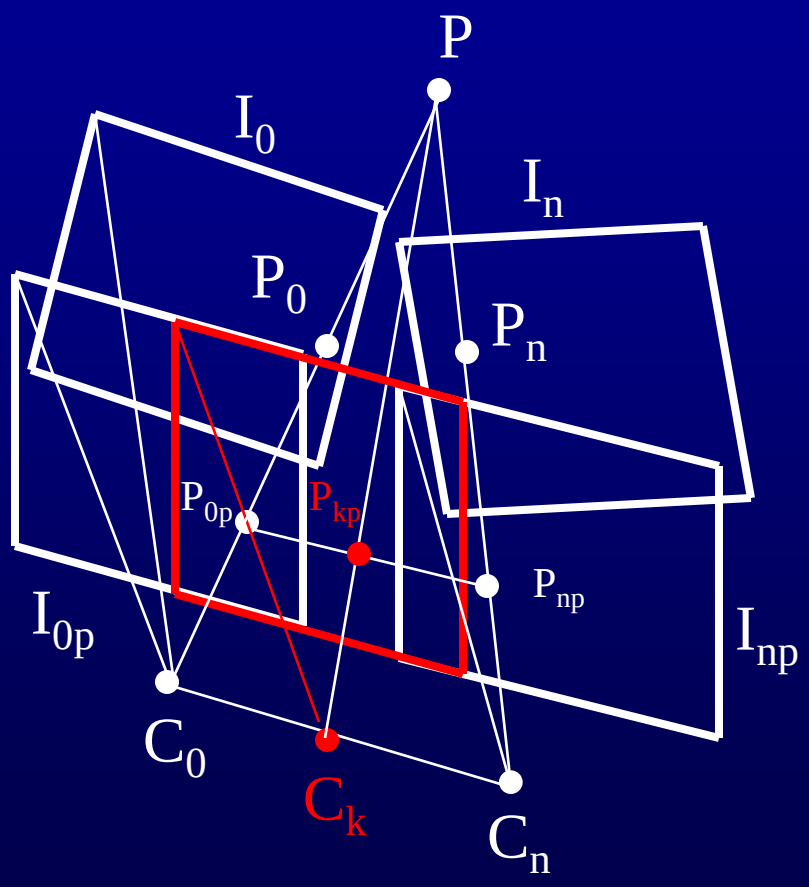
- Shape preserving morph
- Three step algorithm
  1. Prewarp first and last images to parallel views
  2. Image morph between prewarped images
  3. Postwarp to interpolated view

Step 1: prewarp to parallel views



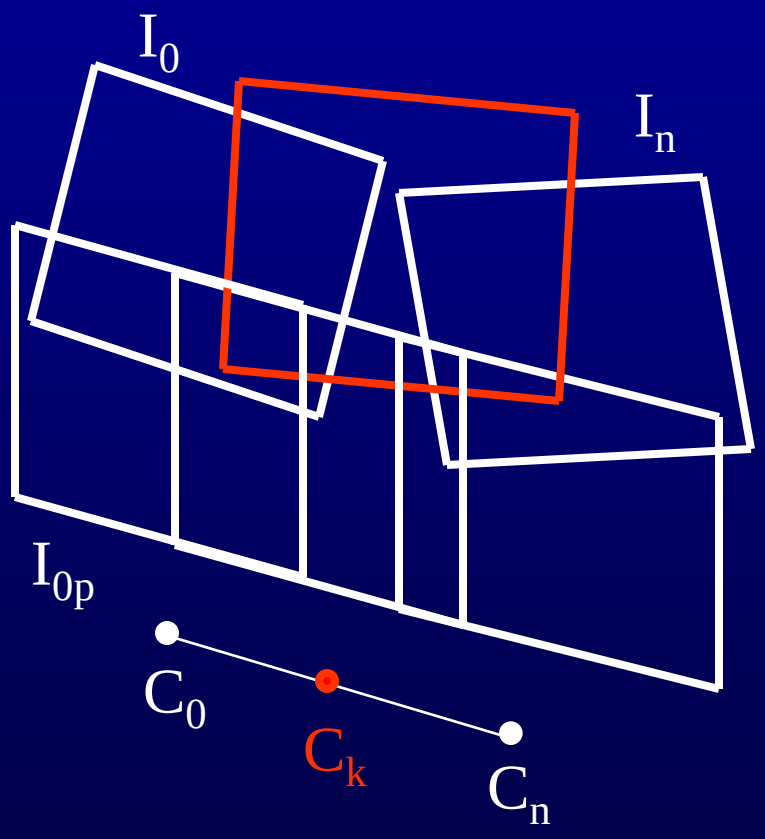
- Parallel views
  - same image plane
  - image plane parallel to segment connecting the two centers of projection
- Prewarp
  - compute parallel views I_{0p}, I_{np}
  - rotate I_0 and I_n to parallel views
  - prewarp corrs. (P_0, P_n) -> (P_{op}, P_{np})

Step 2: morph parallel images



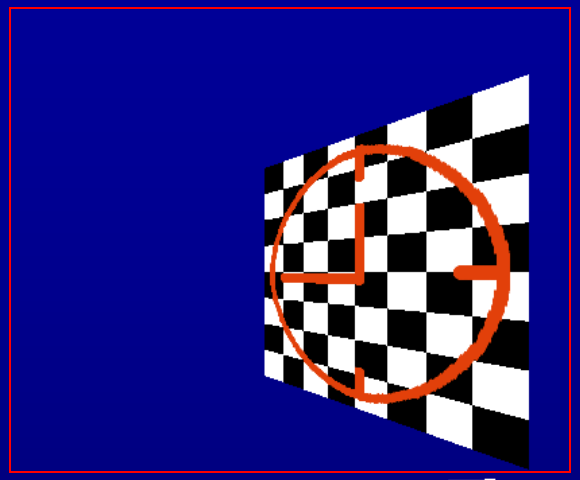
- Shape preserving
- Use prewarped correspondences
- Interpolate C_k from C_0 C_n

Step 3: Postwarping

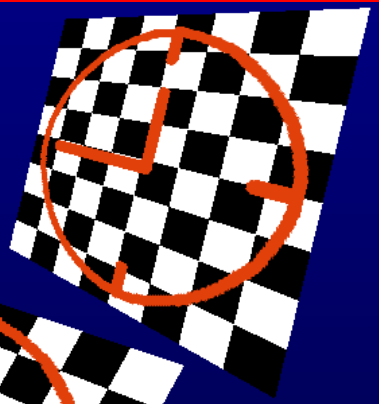


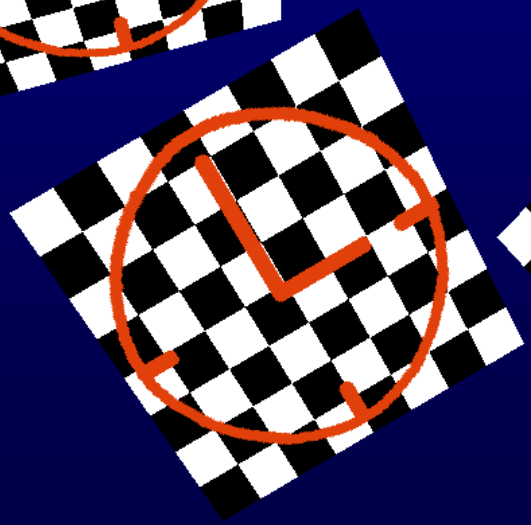
- Postwarp morphed image
  - create intermediate view
    - C_k is known
    - interpolate view direction and tilt
  - rotate morphed image to intermediate view

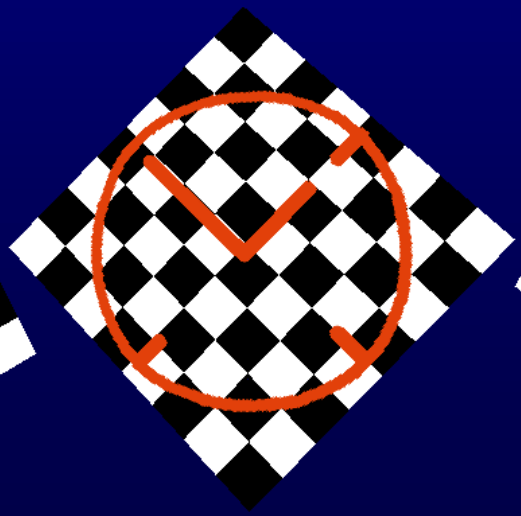
View morphing

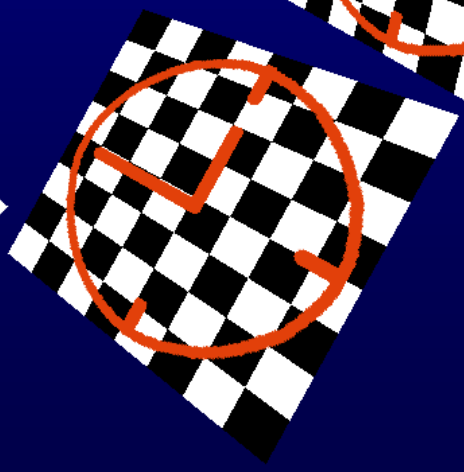


- View morphing
  - shape preserving





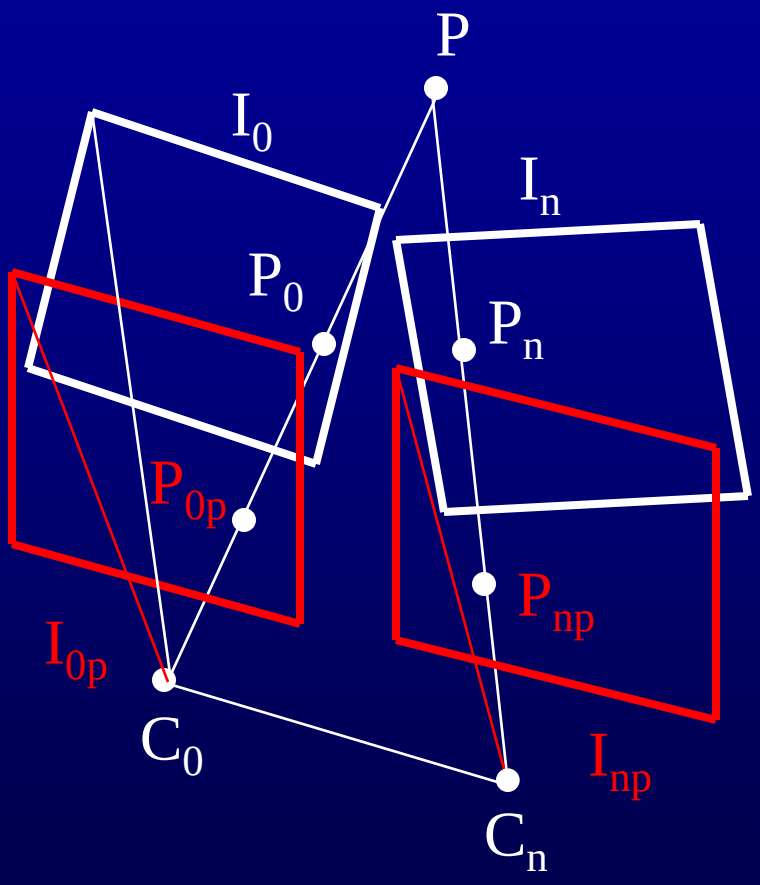




Overview

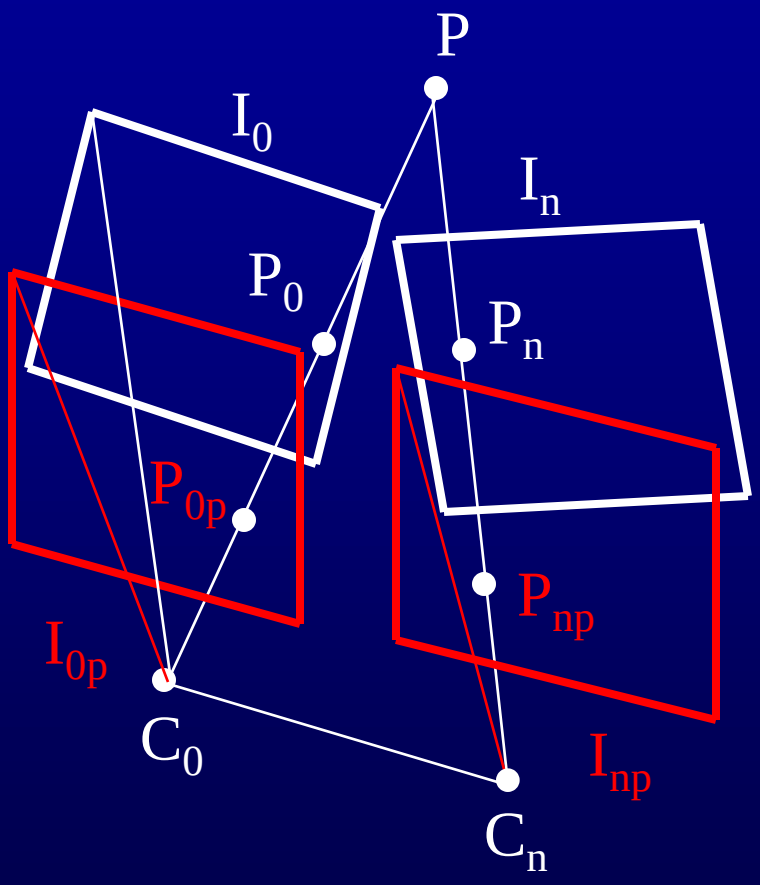
- Introduction
- Image morphing
- View morphing, more details
  - image pre-warping
  - image morphing
  - image post-warping

Step 1: prewarp to parallel views



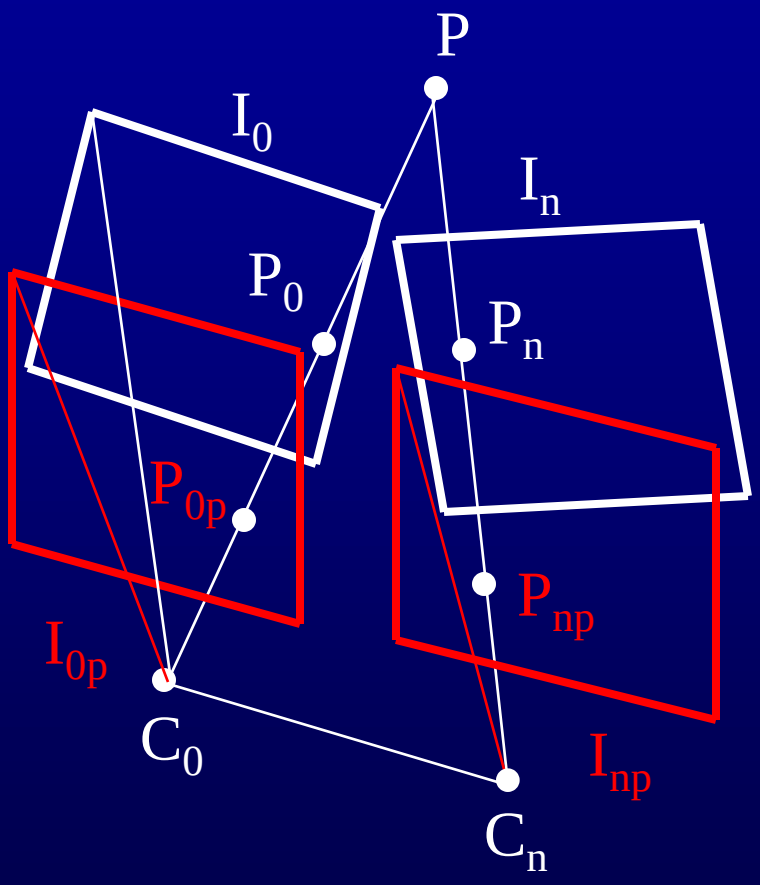
- Parallel views
  - use C_0C_n for x (a_p vector)
  - use (a_0 x b_0) x (a_n x b_n) as y ($-b_p$)
  - pick a_p and b_p to resemble a_0 b_0 as much as possible
  - use same pixel size
  - use wider field of view

Step 1: prewarp to parallel views



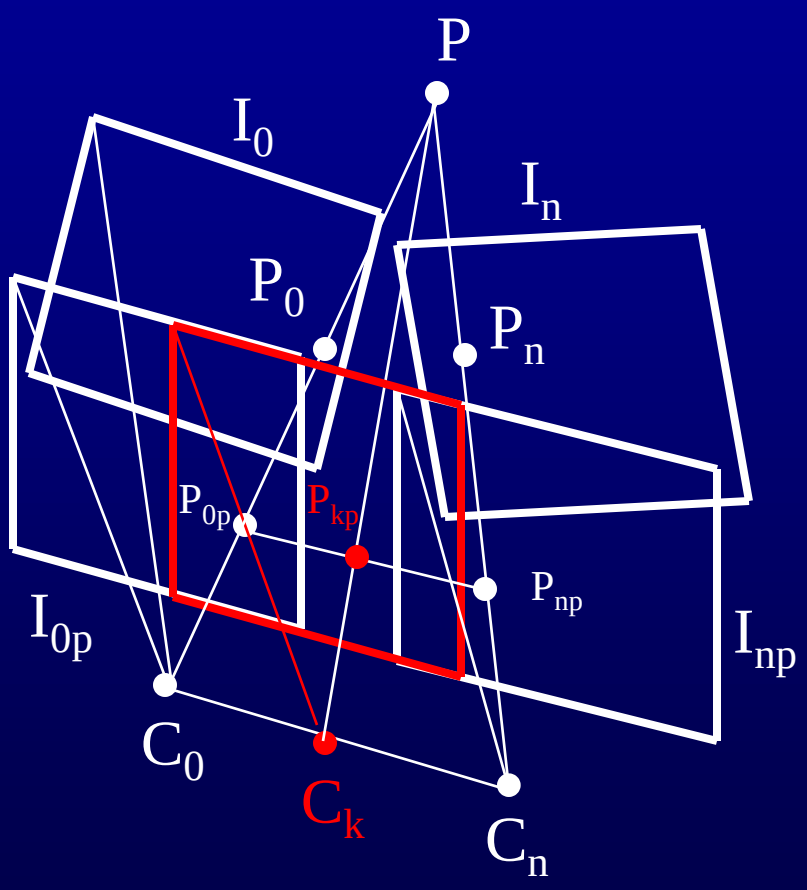
- prewarping using texture mapping
  - create polygon for image plane
  - consider it texture mapped with the image itself
  - render the “scene” from prewarped view
  - if you go this path you will have to implement clipping with the COP plane
  - you have texture mapping already
- alternative: prewarping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

Step 1: prewarp to parallel views



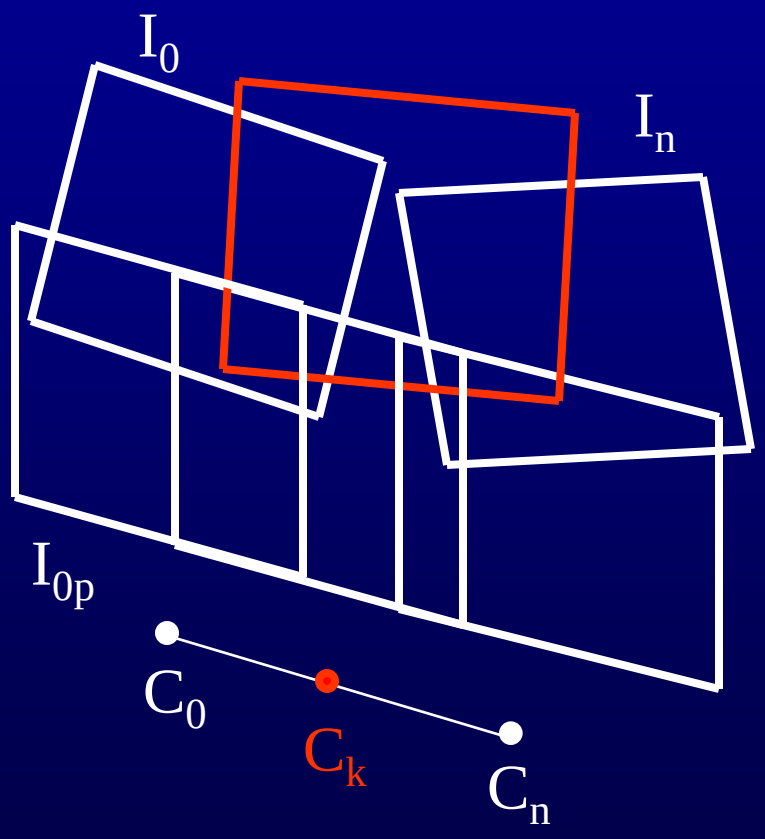
- prewarping correspondences
  - for all pairs of correspondence P_0 P_n
    - project P_0 on I_{0p}, computing P_{0p}
    - project P_n on I_{np}, computing P_{np}
    - prewarped correspondence is P_{op} P_{np}

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
  - useful observation
    - corresponding pixels are on same line in prewarped views
  - preventing holes
    - use larger footprint (ex 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 on disparity
    - holes
      - morph I_{np} to I_{kp}
      - 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