Image Morphing, Warping, and Resizing

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University

Topics

- Image morphing (2D)
  - A “hack”
- View morphing (2D+)
  - Planar warp using homography
- Image warping (3D)
  - Full 3D warp

Motivation – Rendering from Images

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

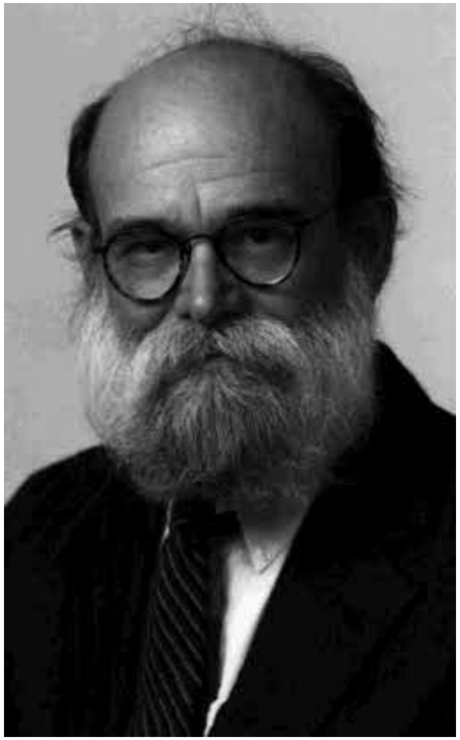
Related Work

- Panoramas ([Chen95/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)

Image Morphing



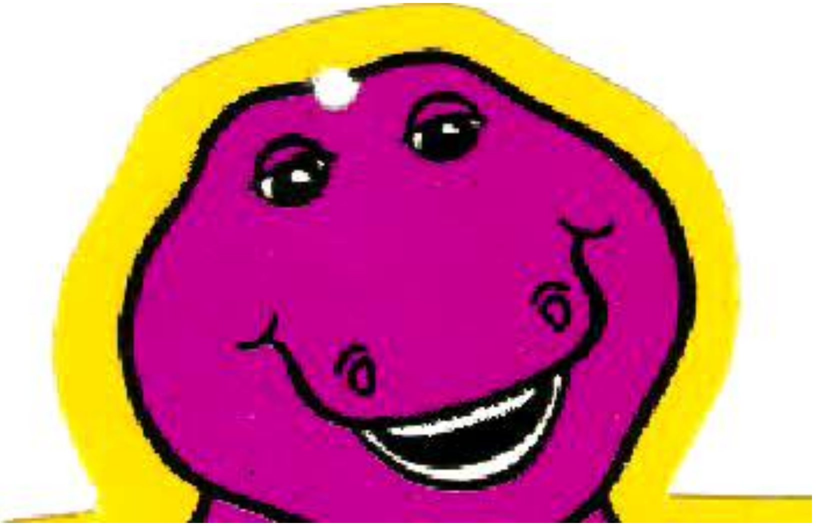


Image Morphing

- 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

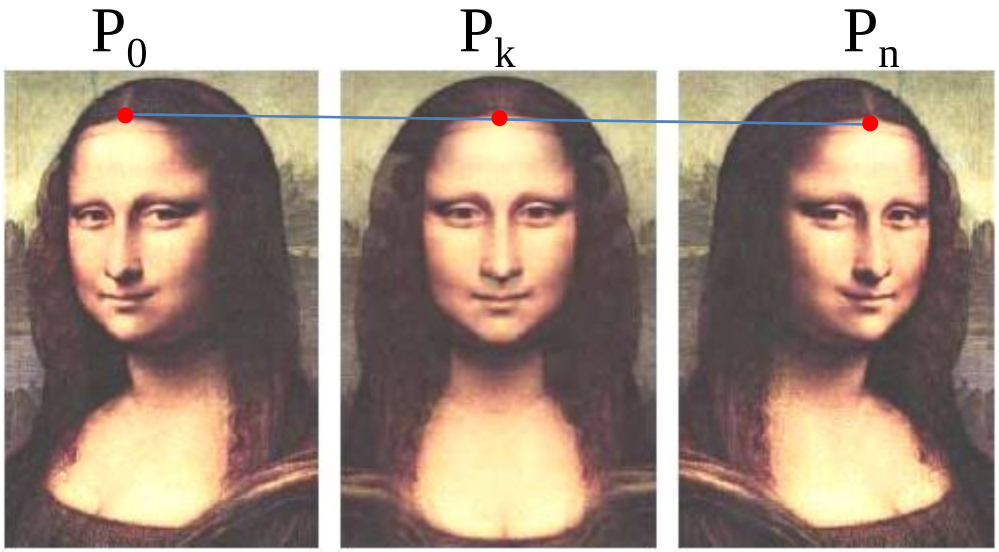
Image Morphing

1. Correspondences

Image Morphing

1. Correspondences

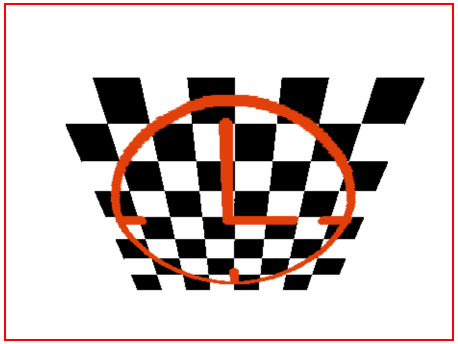
Image Morphing



1. Correspondences
2. Linear interpolation

$$P_k = (1 - \frac{k}{n})P_0 + \frac{k}{n}P_n$$

Image Morphing



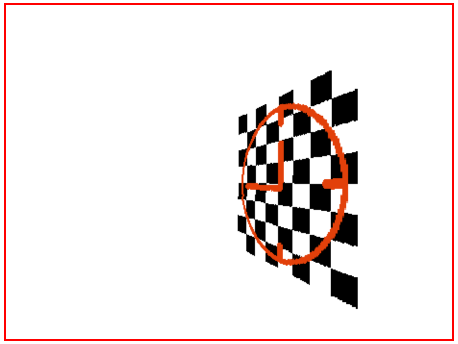


Image Morphing

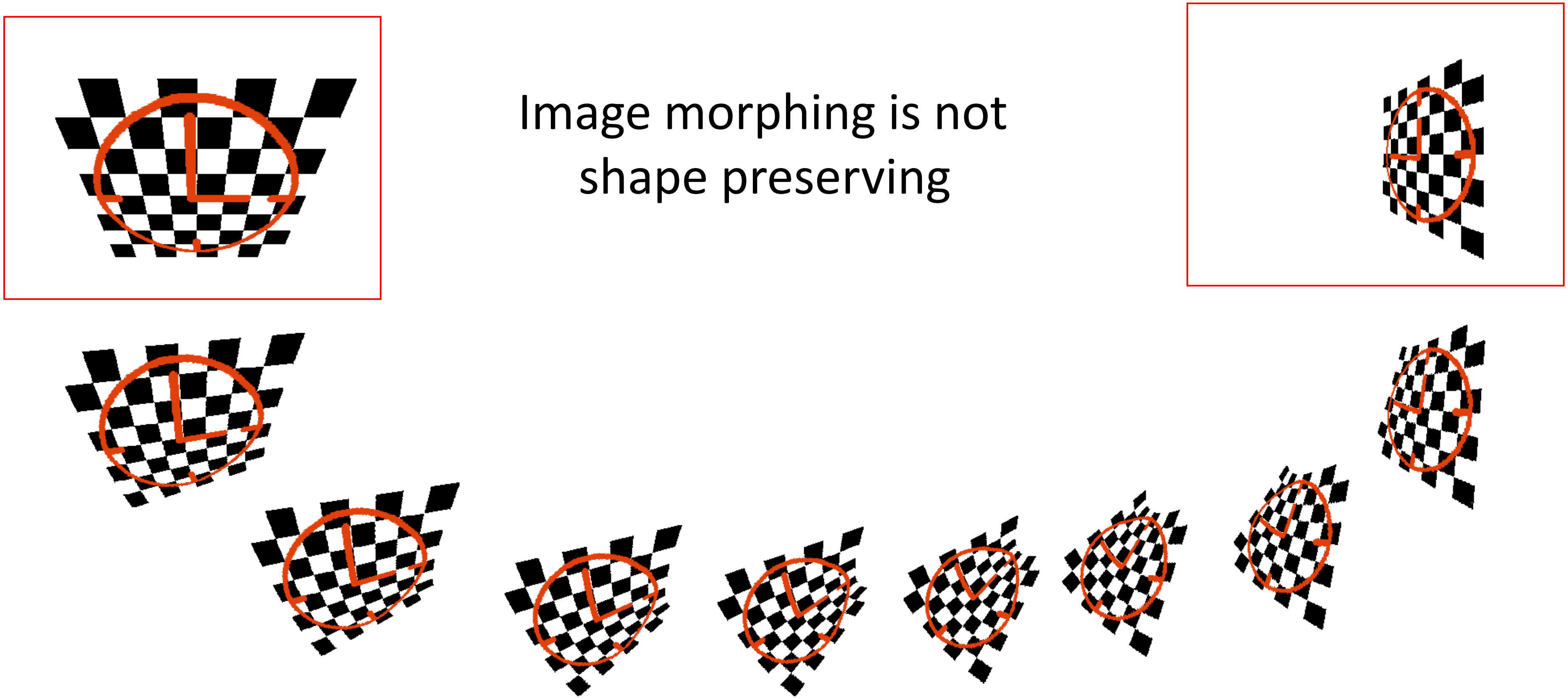
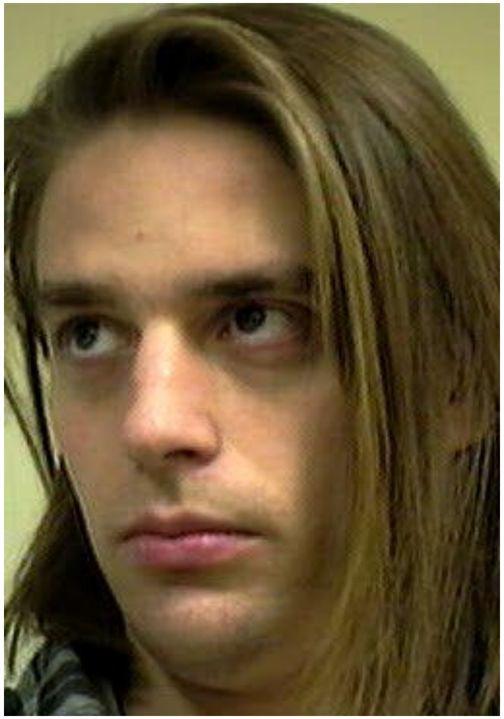


Image morphing is not shape preserving

Topics

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

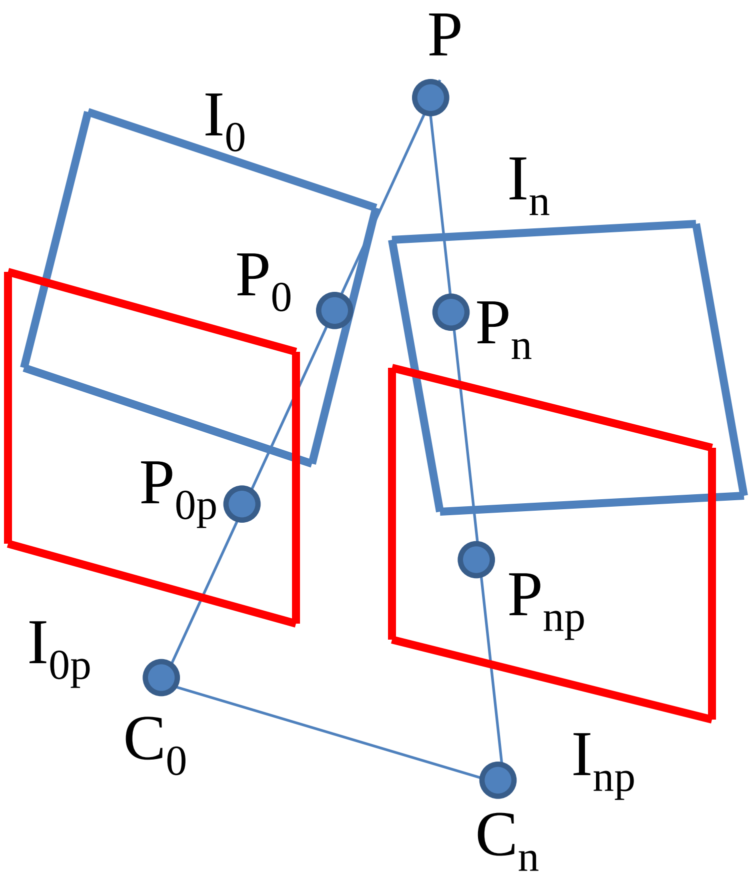
View Morphing



View Morphing

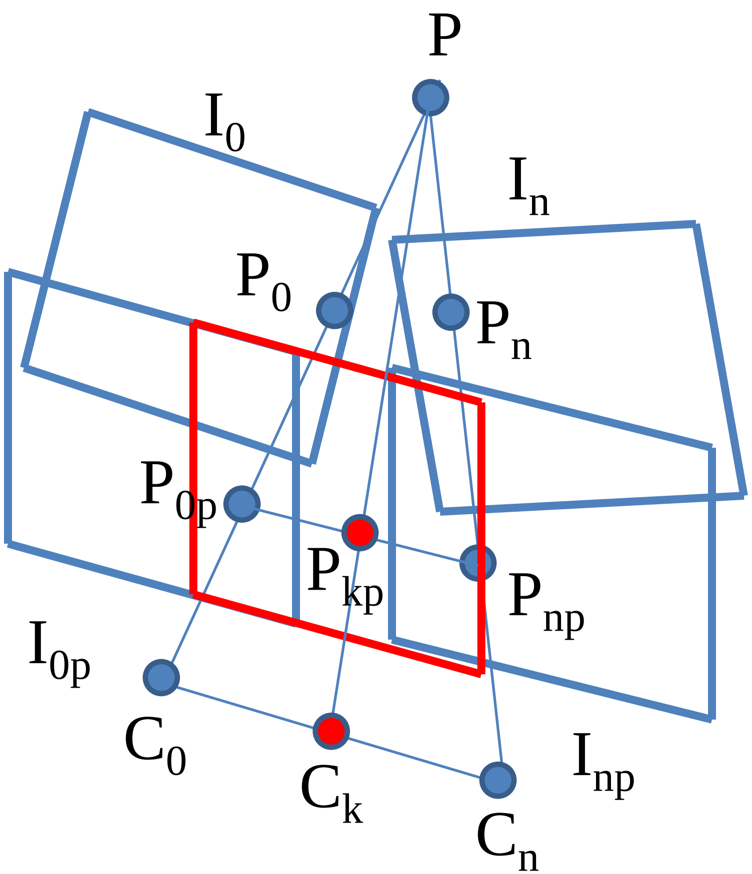
- Shape preserving morph
- Three step algorithm
  - Prewarp first and last images to parallel views
  - Image morph between prewarped images
  - Postwarp to interpolated view

Step 1: prewarp to parallel views



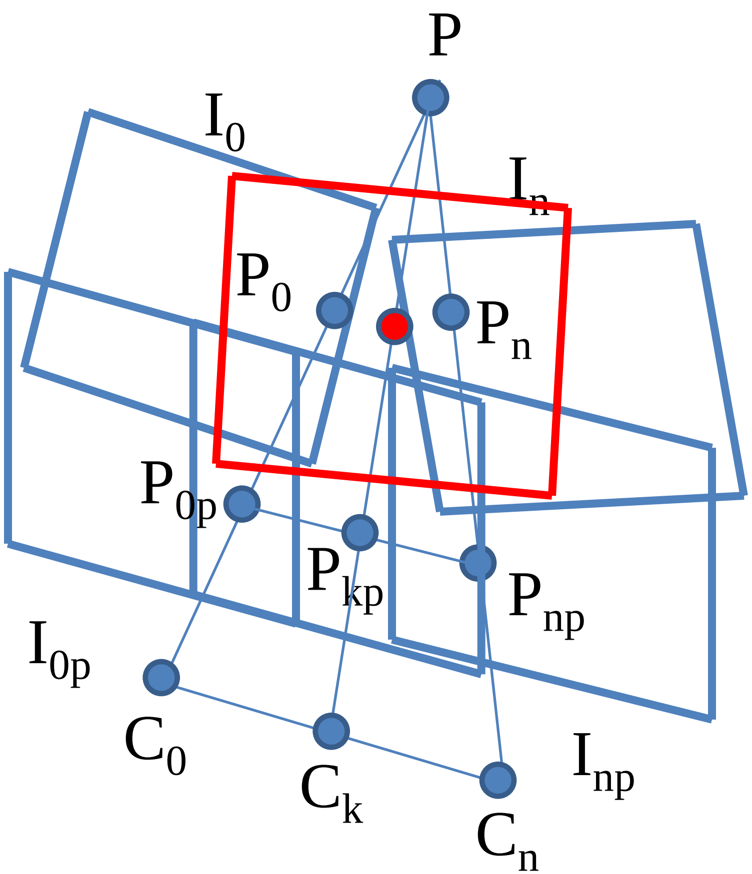
- Parallel views
  - same image plane
  - image plane parallel to segment connecting the two centers of projection
- Prewarp
  - compute parallel views I_{0p}, I_{np}
  - rotate I_0 and I_n to parallel views
  - prewarp correspondence is $(P_0, P_n) \rightarrow (P_{op}, P_{np})$

Step 2: morph parallel images



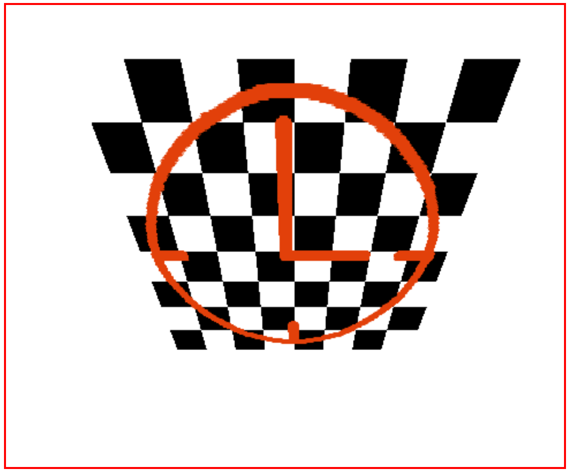
- Shape preserving
- Use prewarped correspondences
- Interpolate C_k from C_0 C_n

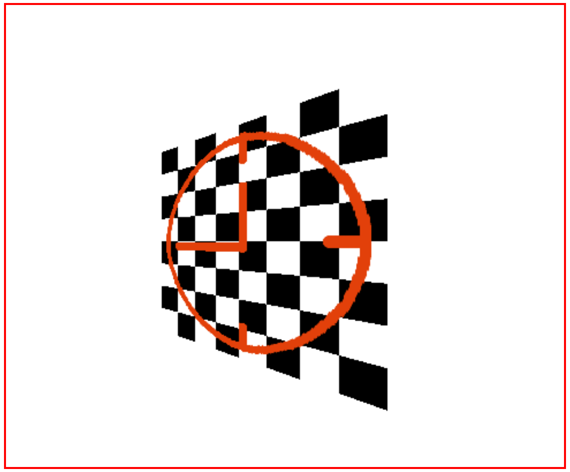
Step 3: postwarp image



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

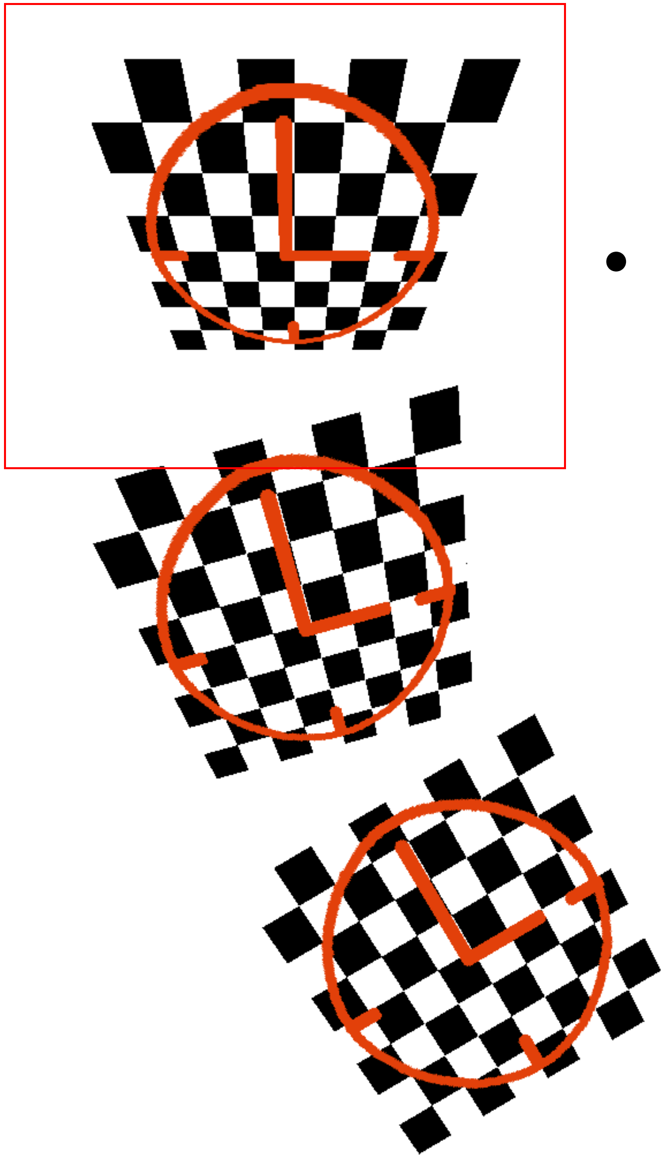
View morphing

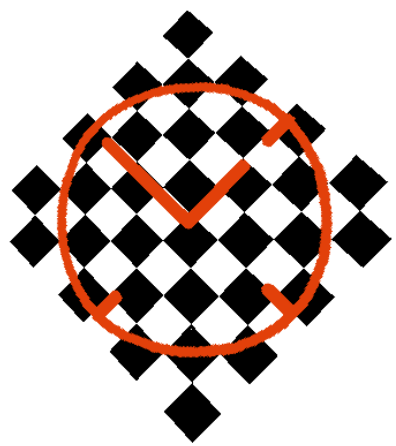


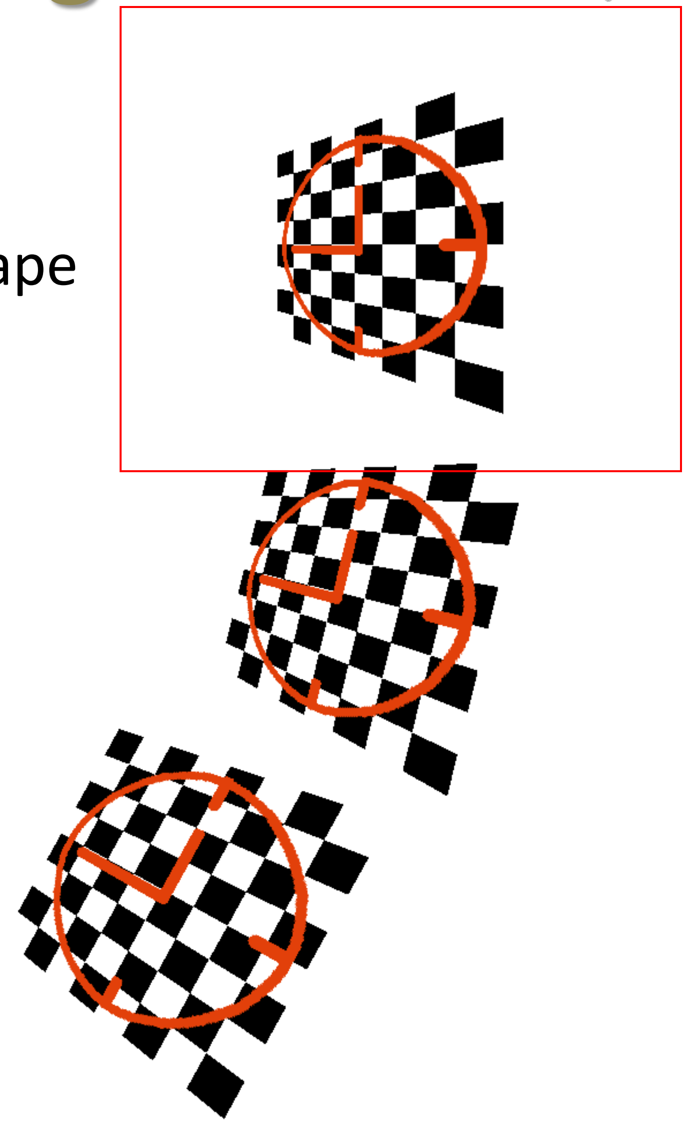


View morphing

- View morphing is shape preserving

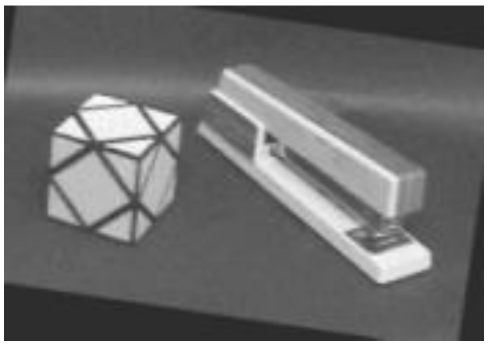
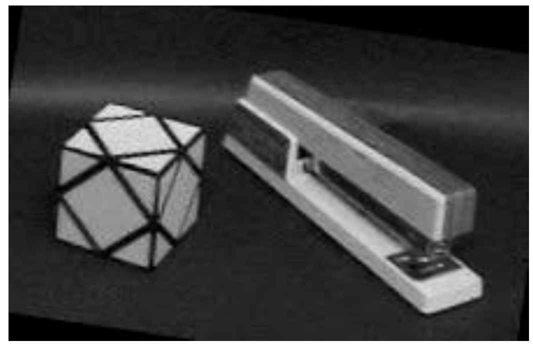
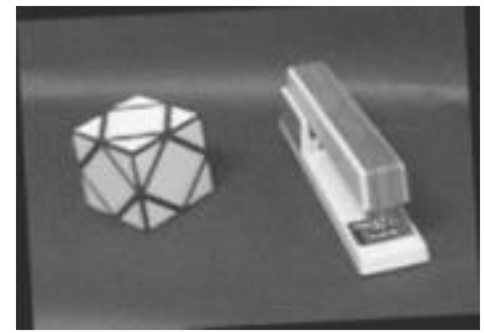






View Morphing Examples

- Using computer vision/stereo reconstruction techniques

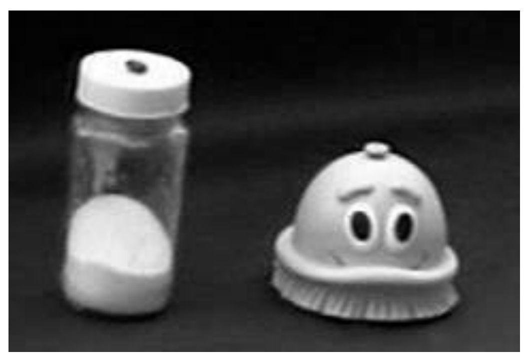
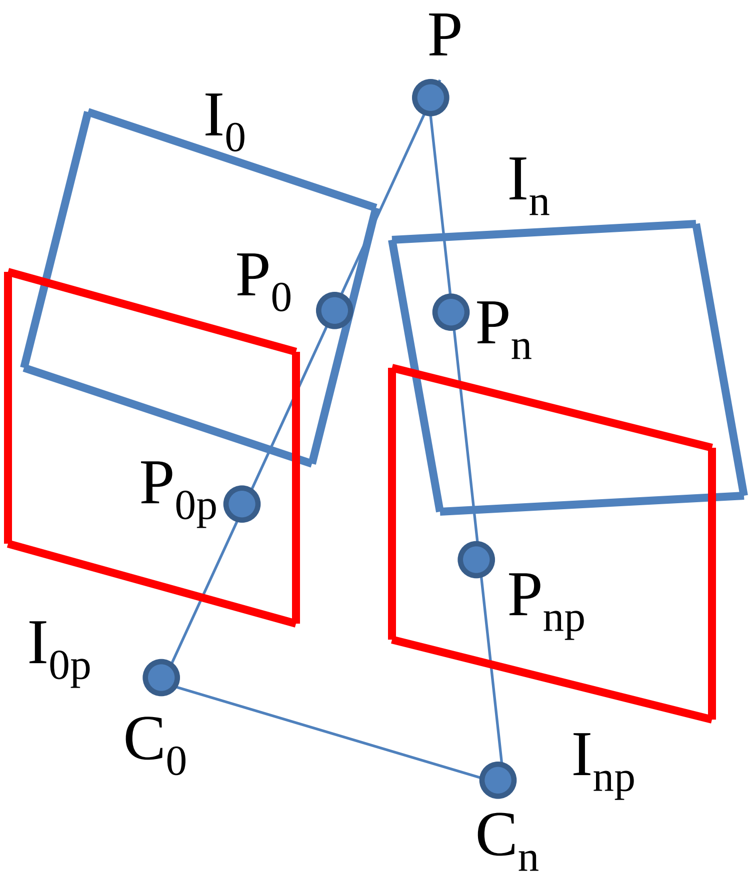




Image Transformations



- 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 ≥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

$$\begin{pmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0 x_0 & -v_0 x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1 x_1 & -v_1 x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2 x_2 & -v_2 x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3 x_3 & -v_3 x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0 y_0 & -v_0 y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1 y_1 & -v_1 y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2 y_2 & -v_2 y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3 y_3 & -v_3 y_3 \end{pmatrix} A = b$$

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)

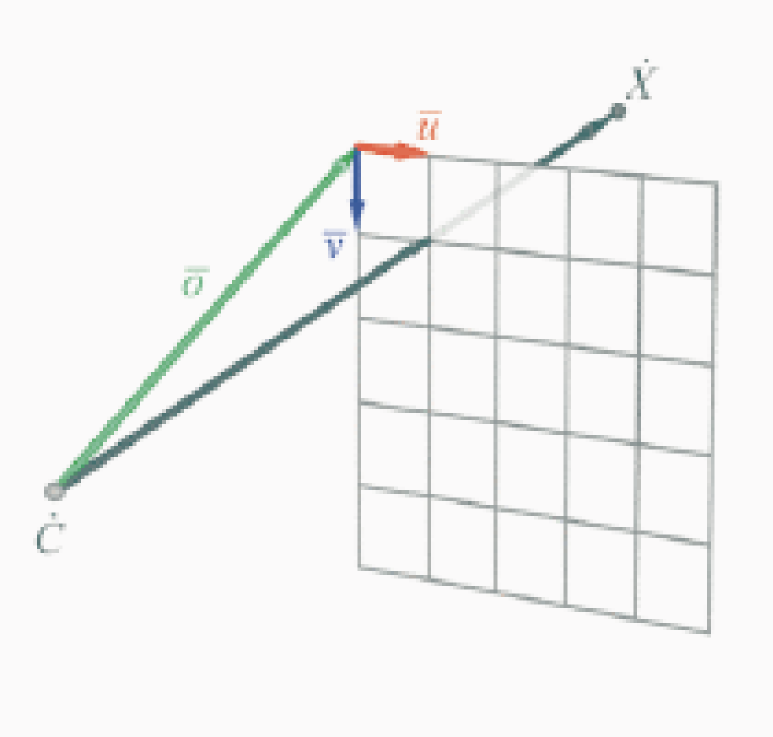
3D Image Warping Equations

McMillan & Bishop Warping Equation:

$$x_2 = \underbrace{\delta(x_1)\, P_2^{-1} (c_1 - c_2)}_{\text{Move pixels based on distance to eye}} + \underbrace{P_2^{-1} P_1\, x_1}_{\text{\textasciitilde Texture mapping}}$$

- Per-pixel distance values are used to warp pixels to their correct location for the current eye position

3D Image Warping Equations

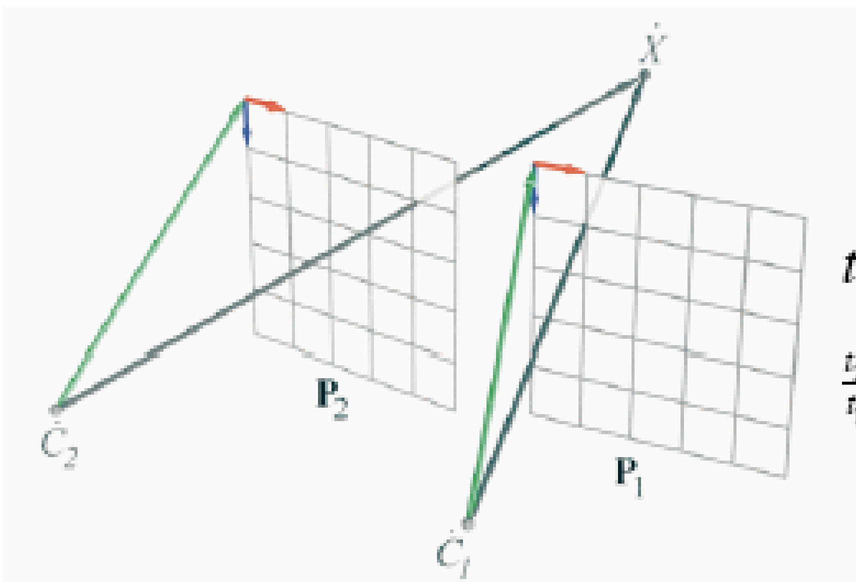


$$P = \begin{bmatrix} u_x & v_x & o_x \\ u_y & v_y & o_y \\ u_z & v_z & o_z \end{bmatrix}$$

$$\dot{X} = \dot{C} + t\,P\,\vec{x}$$

Some pictures courtesy of SIGGRAPH '99 course notes
(Leonard McMillan)

3D Image Warping Equations



$$\dot{C}_2 + t_2 P_2 \vec{x}_2 = \dot{C}_1 + t_1 P_1 \vec{x}_1$$

$$t_2 P_2 \vec{x}_2 = \dot{C}_1 - \dot{C}_2 + t_1 P_1 \vec{x}_1$$

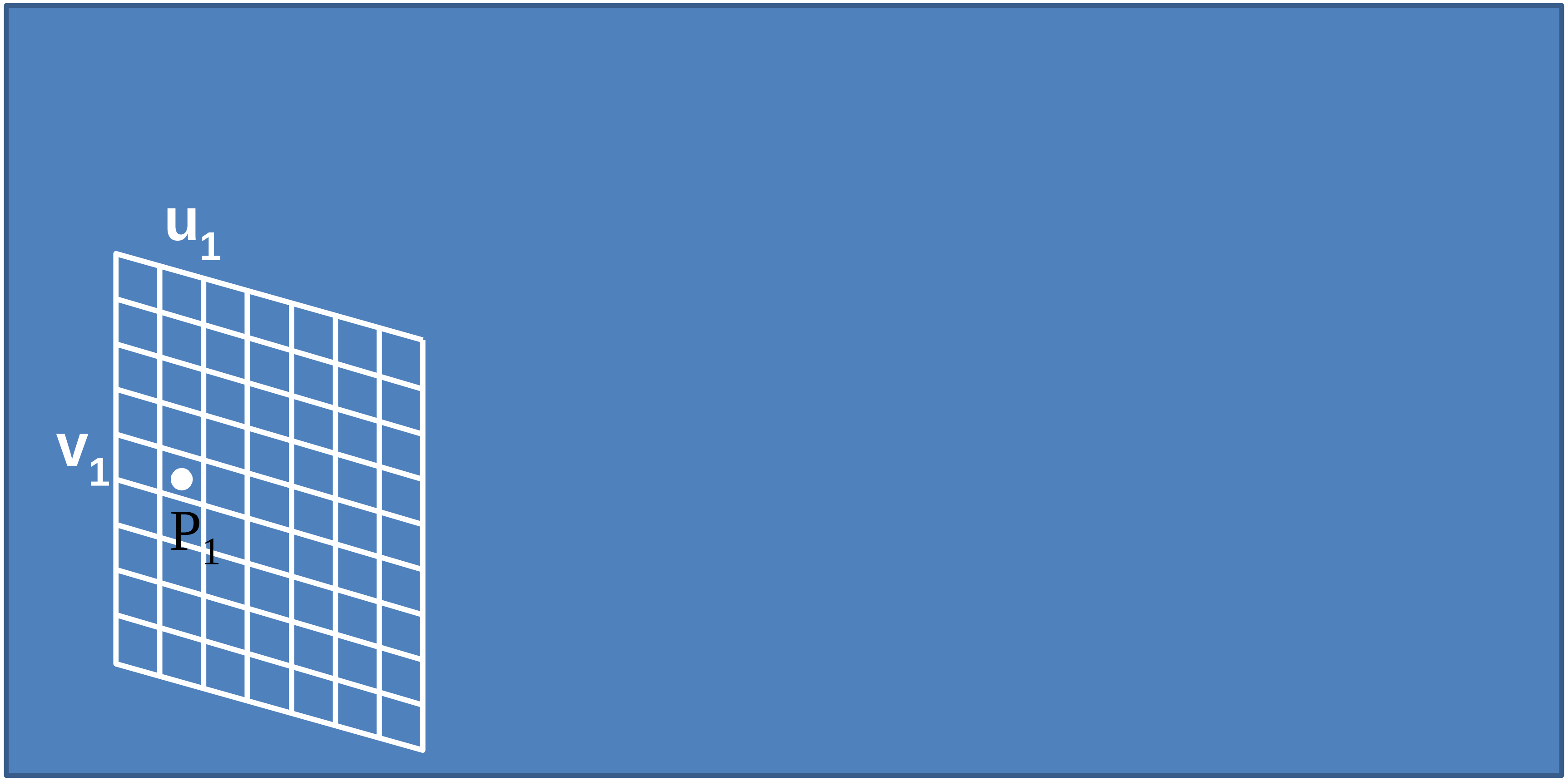
$$t_2 \vec{x}_2 = P_2^{-1}\left(\dot{C}_1 - \dot{C}_2\right) + t_1 P_2^{-1} P_1 \vec{x}_1$$

$$\tfrac{t_2}{t_1} \vec{x}_2 = \tfrac{1}{t_1} P_2^{-1}\left(\dot{C}_1 - \dot{C}_2\right) + P_2^{-1} P_1 \vec{x}_1$$

$$\vec{x}_2 \doteq \underbrace{\tfrac{1}{t_1}}_{\delta} \underbrace{P_2^{-1}\left(\dot{C}_1 - \dot{C}_2\right)}_{e_{21}} + \underbrace{P_2^{-1} P_1}_{H_{21}} \vec{x}_1$$

3D Image Warping Equations

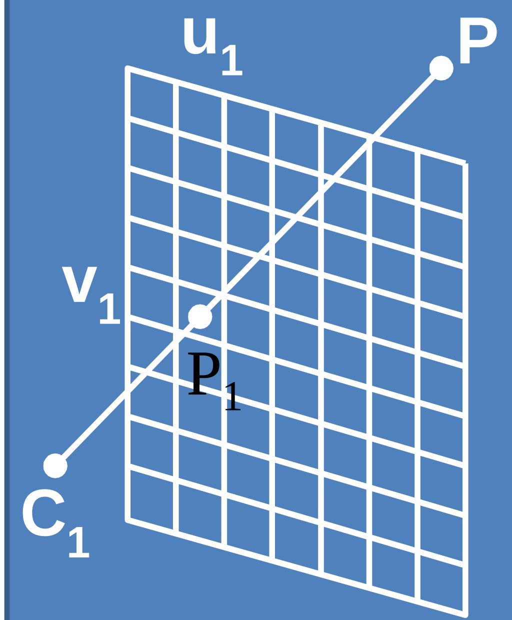
- Images enhanced with per-pixel depth [McMillan95]



3D Image Warping Equations

$$\dot{P} = \dot{C}_1 + (\bar{c}_1 + u_1 \bar{a}_1 + v_1 \bar{b}_1) w_1$$

$$w_1 = \frac{C_1 P}{C_1 P_1}$$

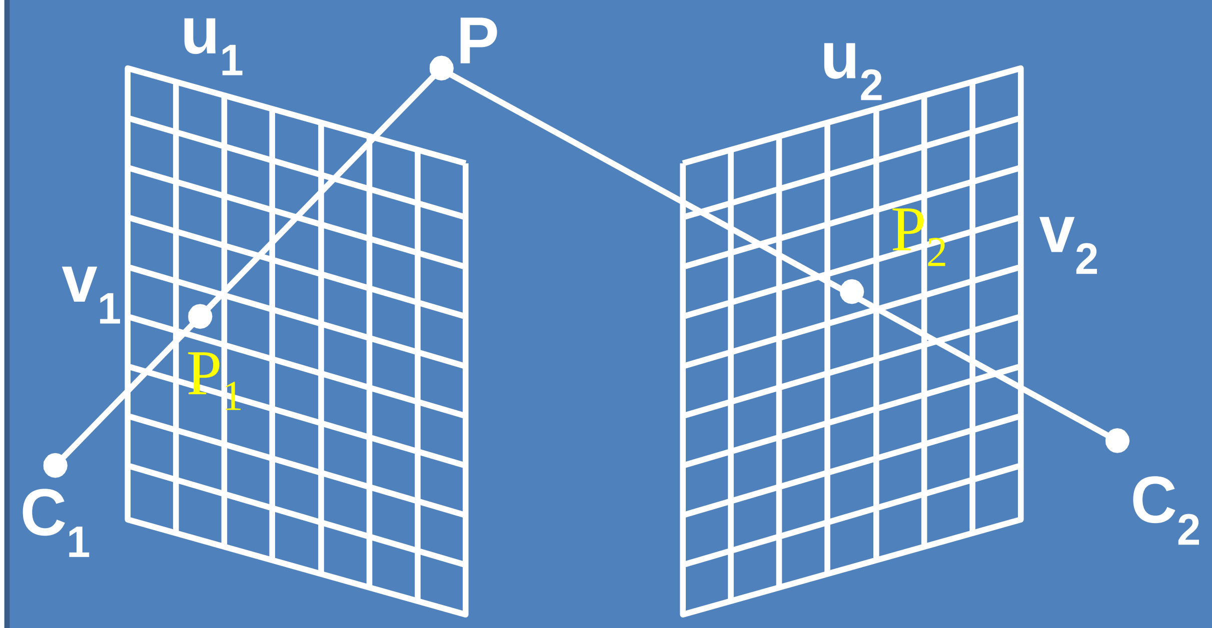


- $1/w_1$ also called generalized disparity
- another notation $\delta(u_1, v_1)$

3D Image Warping Equations

$$\dot{P} = \dot{C}_1 + (\bar{c}_1 + u_1 \bar{a}_1 + v_1 \bar{b}_1) w_1$$

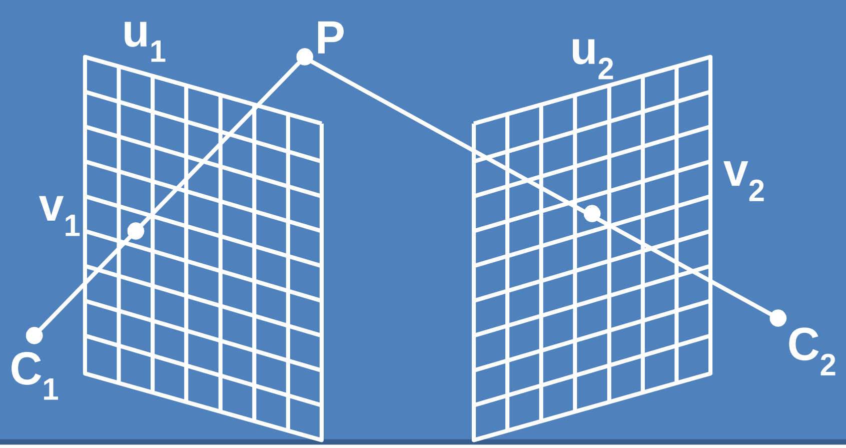
$$\dot{P} = \dot{C}_2 + (\bar{c}_2 + u_2 \bar{a}_2 + v_2 \bar{b}_2) w_2$$



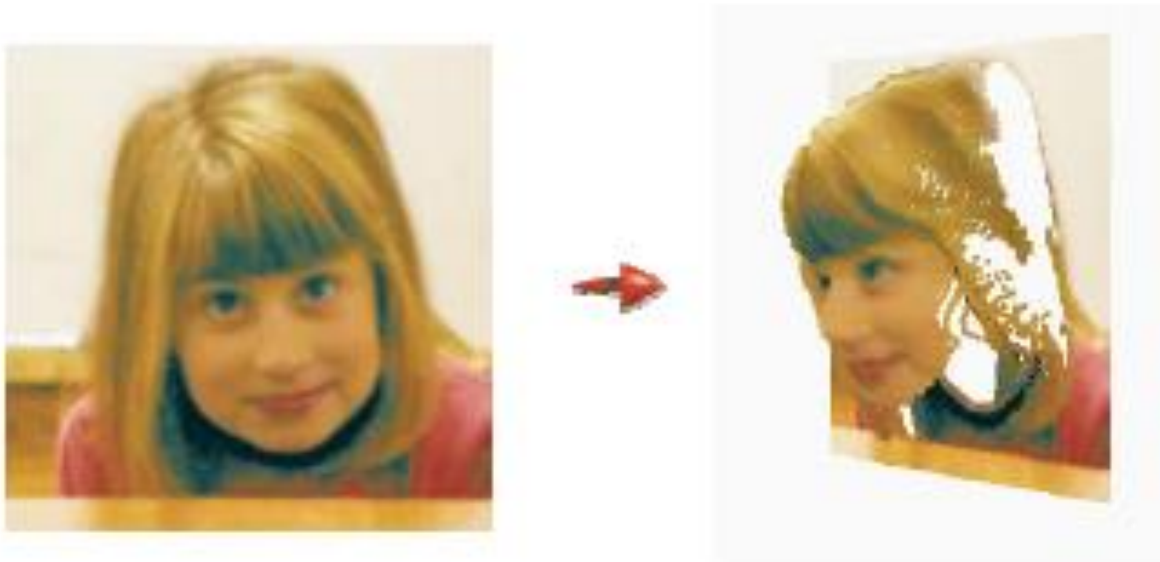
3D Image Warping Equations

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

$$v_2 = \frac{w_{21} + w_{22} \cdot u_1 + w_{23} \cdot v_1 + w_{24} \cdot \delta(u_1, v_1)}{w_{31} + w_{32} \cdot u_1 + w_{33} \cdot v_1 + w_{34} \cdot \delta(u_1, v_1)}$$



3D Image Warping Example



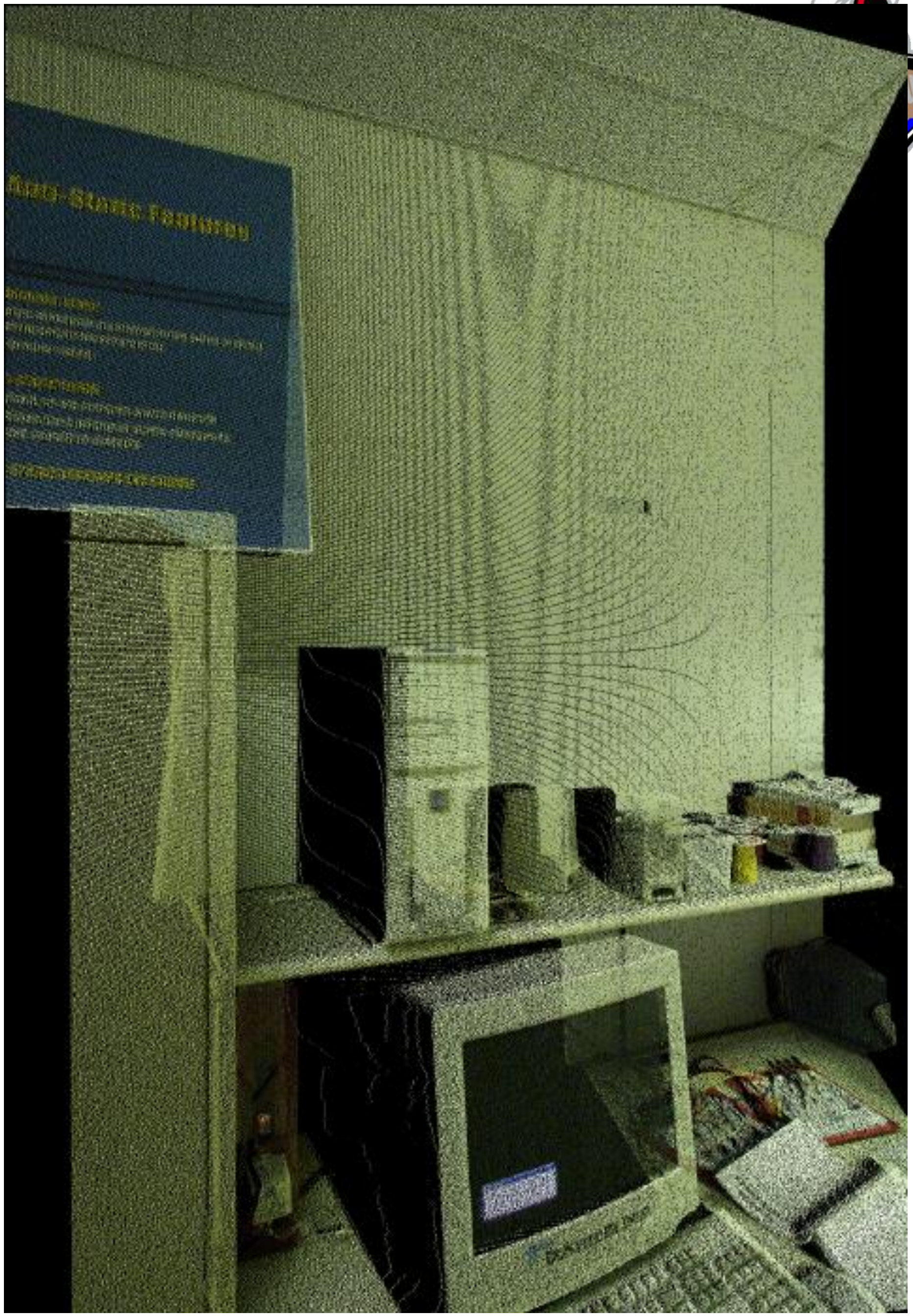
3D Image Warping Example

- 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?

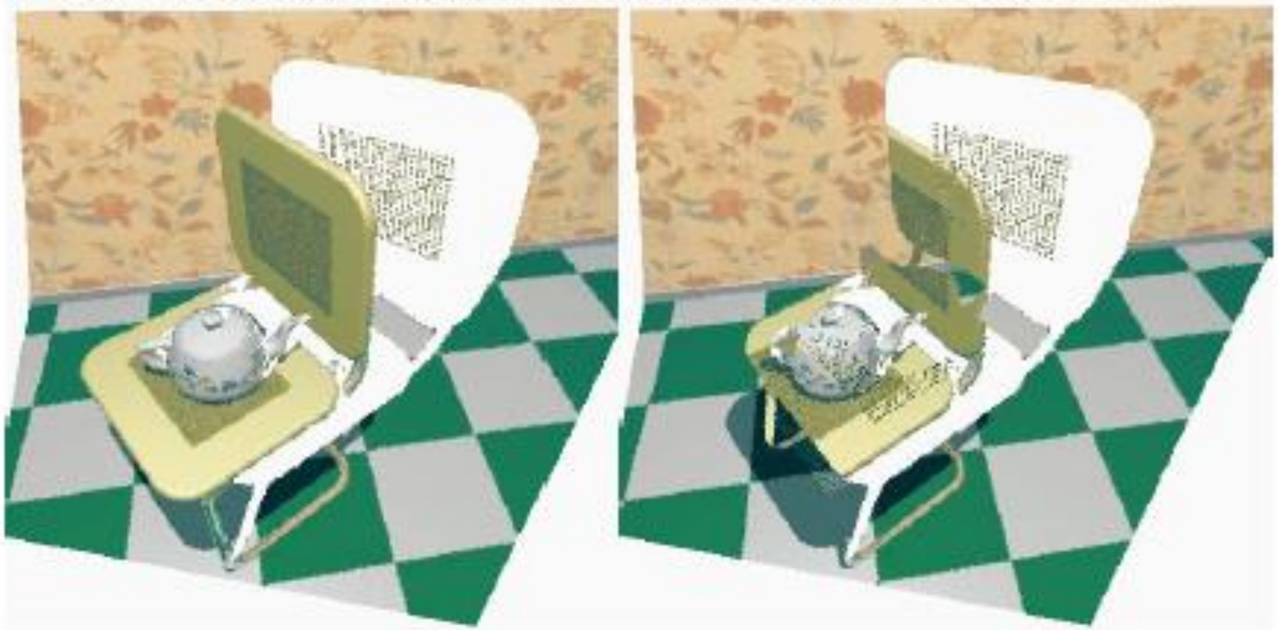
Disocclusions

- Bilinear patches: fill in the areas

What else?

Rendering Order

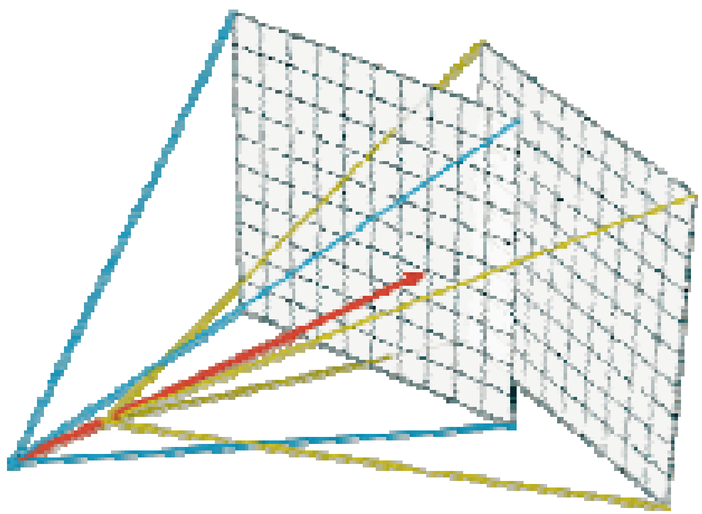
- The warping equation determines where points go…

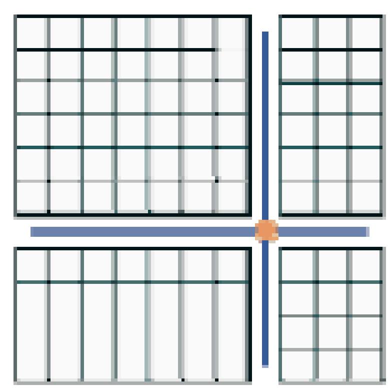


… but that is not sufficient

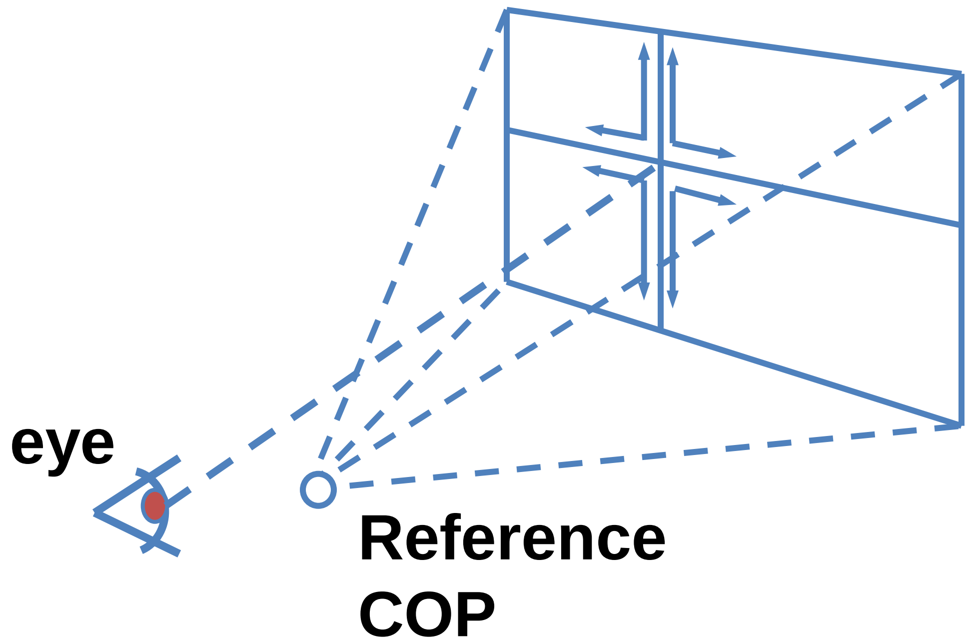
Occlusion Compatible Rendering Order

- 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

The Image Resizing/Retargeting Problem

Image Retargeting Objectives

1. Change size

2. Preserve the important ***content*** and ***structures***
3. Limit ***artifacts*** created

Traditional Methods

Scaling

Cropping

- Scaling introduces distortions
- Cropping removes important parts

Many Existing Resizing Methods

Seam Carving Method

- "Seam Carving for Content-Aware Image Resizing," S. Avidan and A. Shamir
- In Photoshop called "content aware scaling"
- Main idea: **Remove the least noticeable pixels**
  - How? Define an "energy function" that measures how perceptually noticeable each pixel is
- **Remove the pixels with "low energy" and avoid removing pixels with "high energy"**
  - How? Define a criterion for picking which pixels to remove

Possible Energy Functions

- Edgeness
  - Gradient magnitude

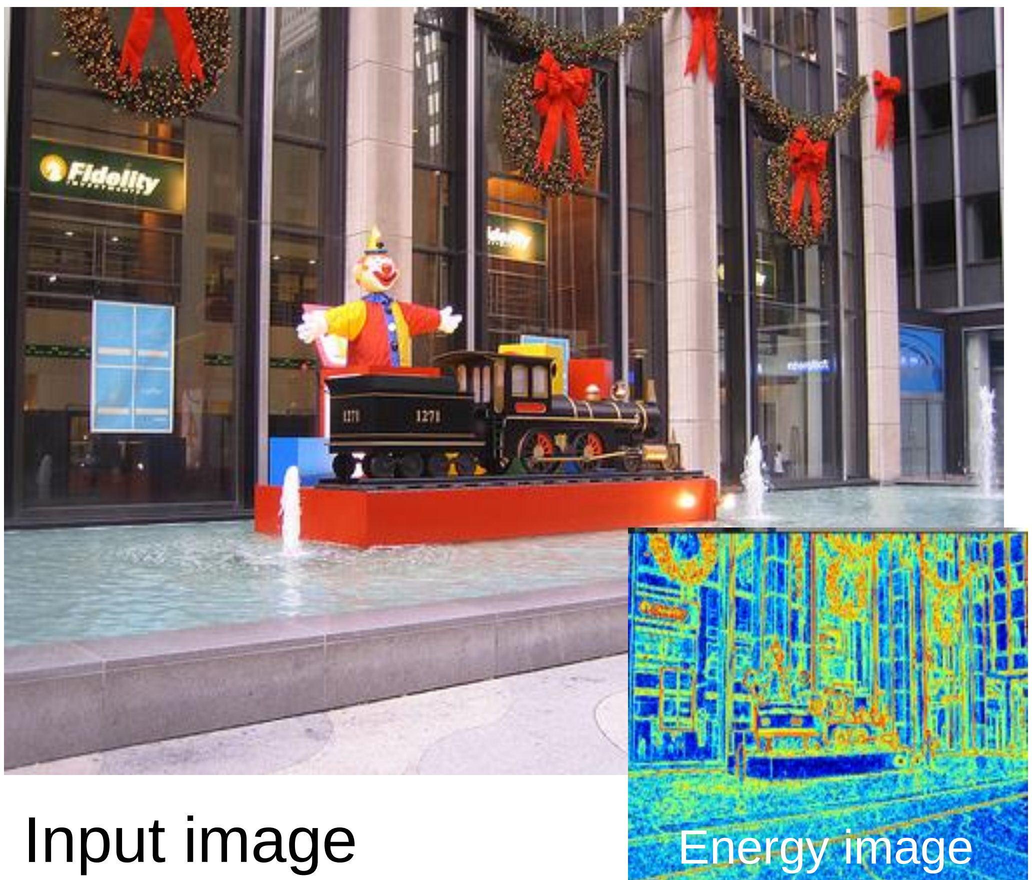
$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

- Entropy
- HOG (Histogram of Gradient)
- Saliency
- …

Caviet: No single energy function performs well on ***all*** images

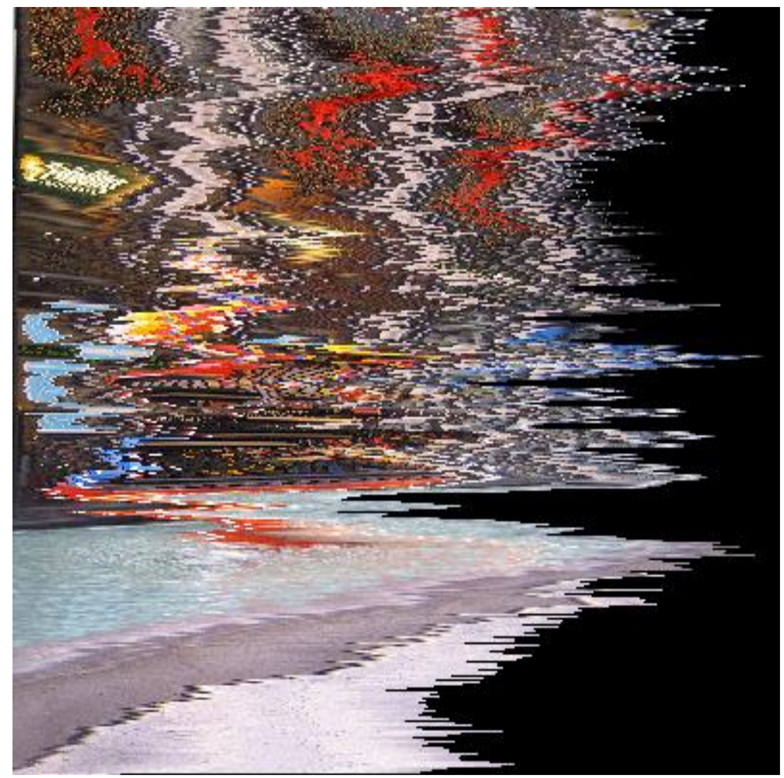
Pixel Removal Criteria

- **Optimal**: remove the *k* pixels with lowest energy
- Output image no longer rectangular



Input image

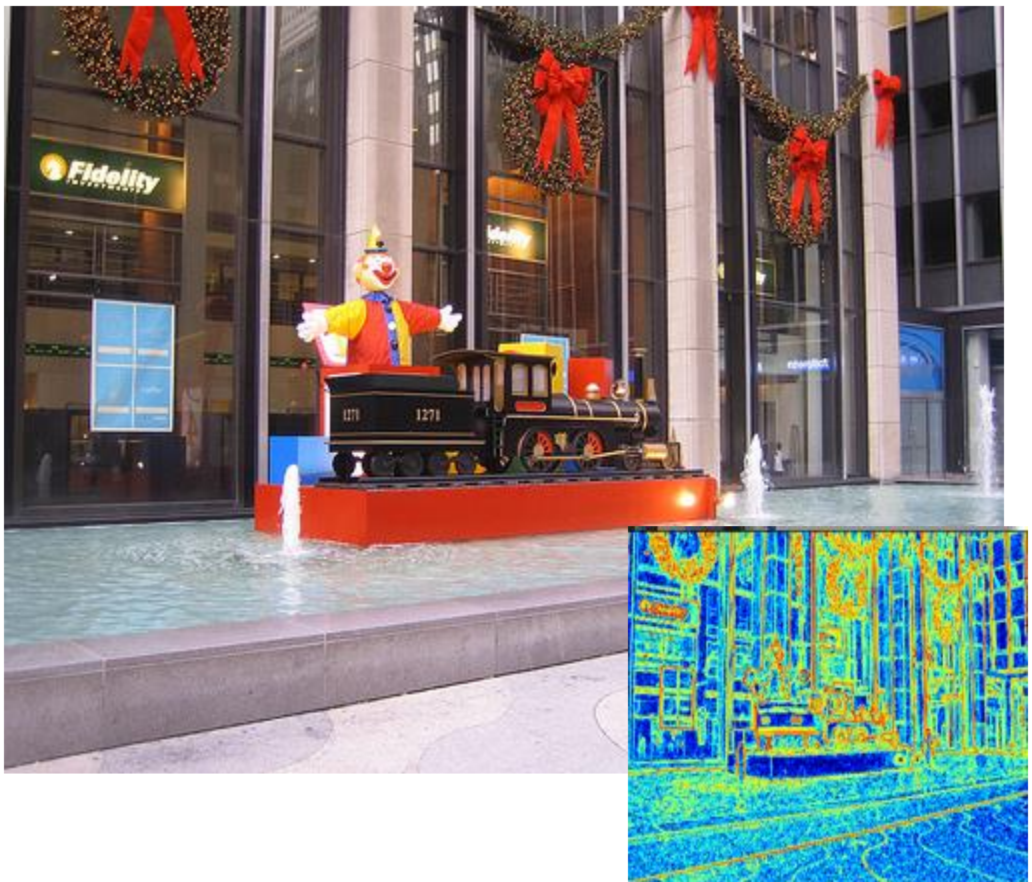
Energy image



Output image

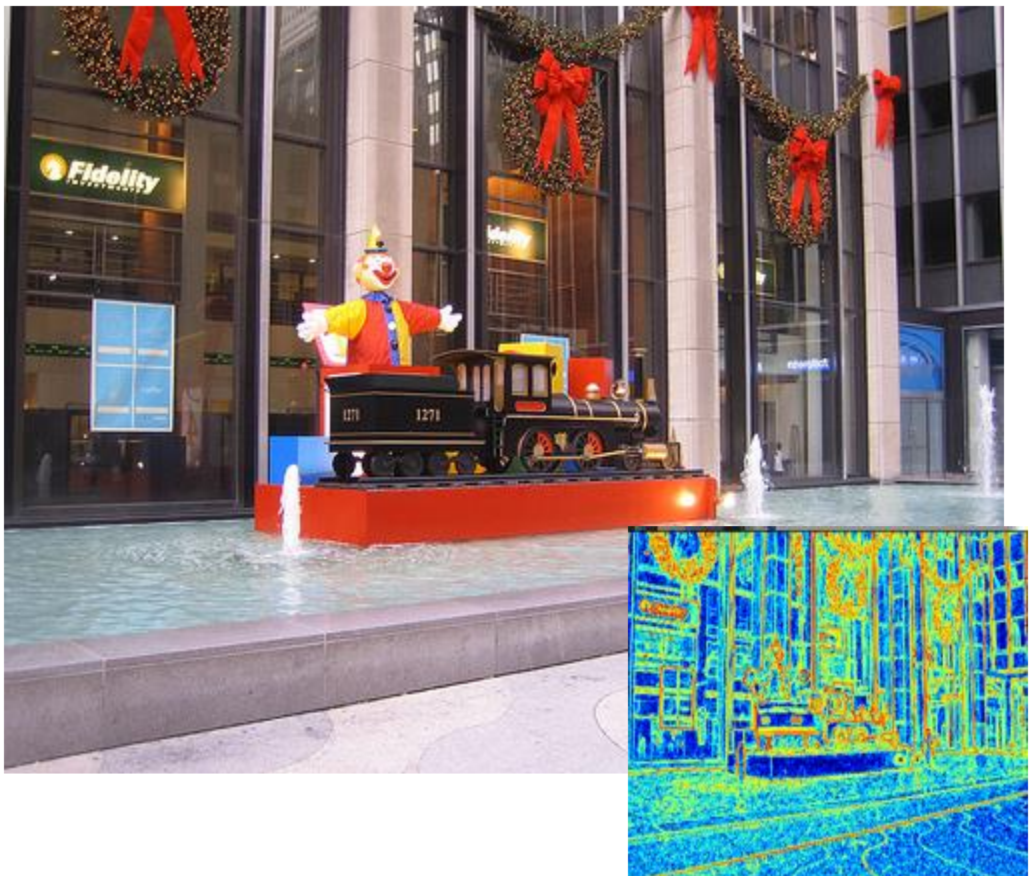
Pixel Removal Criteria

- **Pixel**: Remove *k* pixels with lowest energy in *each* row
- No visual coherence between adjacent rows



Pixel Removal Criteria

- **Column**: Remove whole column with lowest energy
- Frequently introduces artifacts



Seam Definition

- Vertical Seam

  is an 8-connected path of pixels in an *n* x *m* image from top to bottom, containing one, and only one, pixel in each row of the image:

$$\mathbf{s}^{\mathbf{x}} = \{s_i^x\}_{i=1}^{n} = \{(x(i), i)\}_{i=1}^{n}, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1$$

Seam Energy

- Energy of a Seam

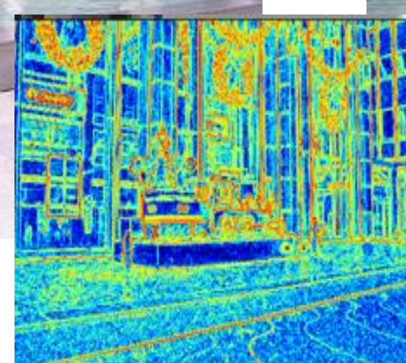
$$E(\mathbf{s}) = E(\mathbf{I_s}) = \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

- Minimum Energy Seam

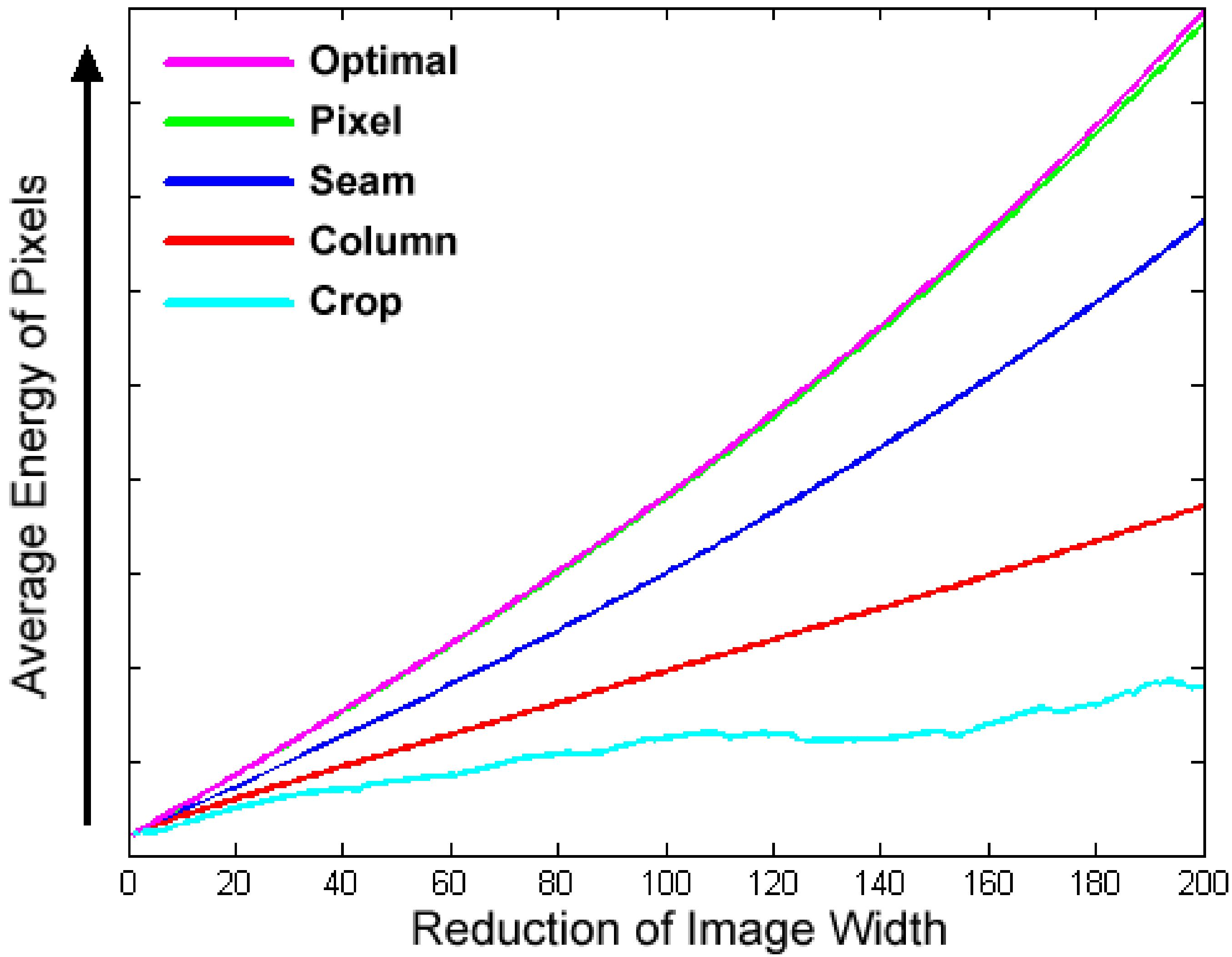
$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

Pixel Removal Criteria

- **Seam**: Remove the vertical curve of lowest energy



Pixel Removal Effectiveness

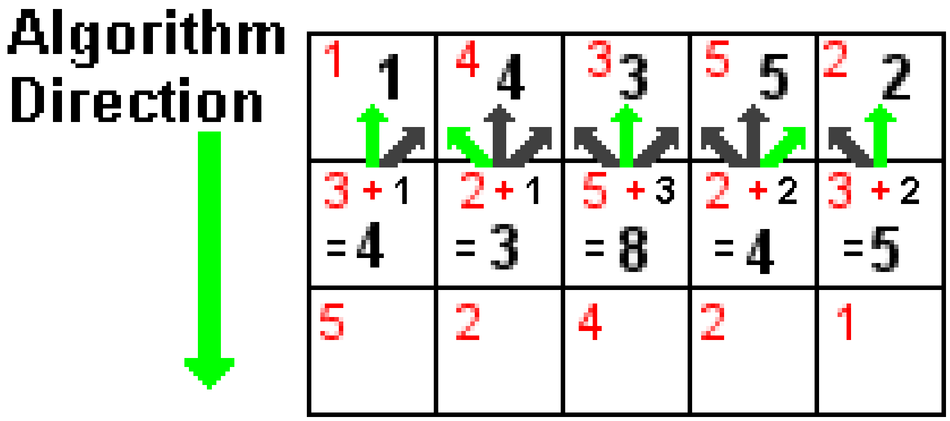


How to Efficiently Compute Best Seam?

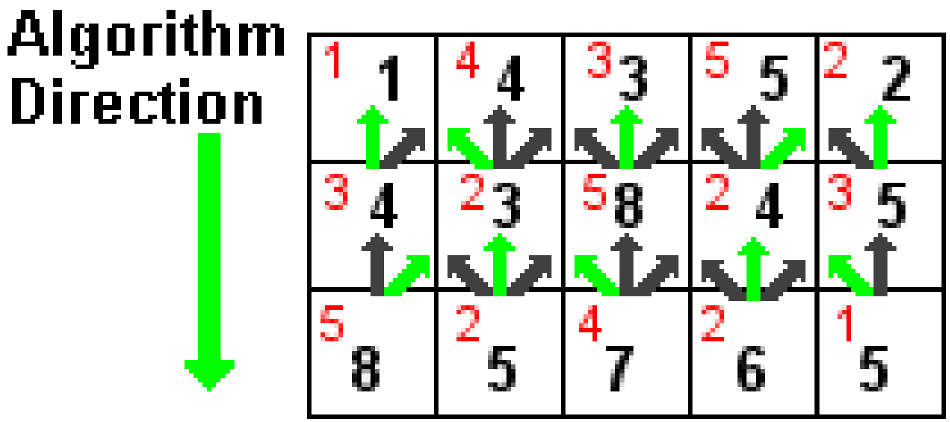
- Use **Dynamic Programming** to find lowest energy seam in linear time

1. **Forward Pass** (top row to bottom row for finding vertical seam)
   - Define $M(i, j)$ = total energy of path ending at (i, j)
   - $M(1, j) = e(1, j)$
   - $M(i, j) = e(i, j) + \min(M(i\text{-}1, j\text{-}1), M(i\text{-}1, j), M(i\text{-}1, j\text{+}1))$
   - $B(i, j) = \mathrm{argmin}_{k=j-1,\, j,\, j+1}\ M(i\text{-}1, k)$
   - Find minimum value in last row: $\min_j M(n, j)$
2. **Backward Pass** (bottom row to top row)
   - Trace back path from pixel in bottom row with min value to top row using B

Forward Pass

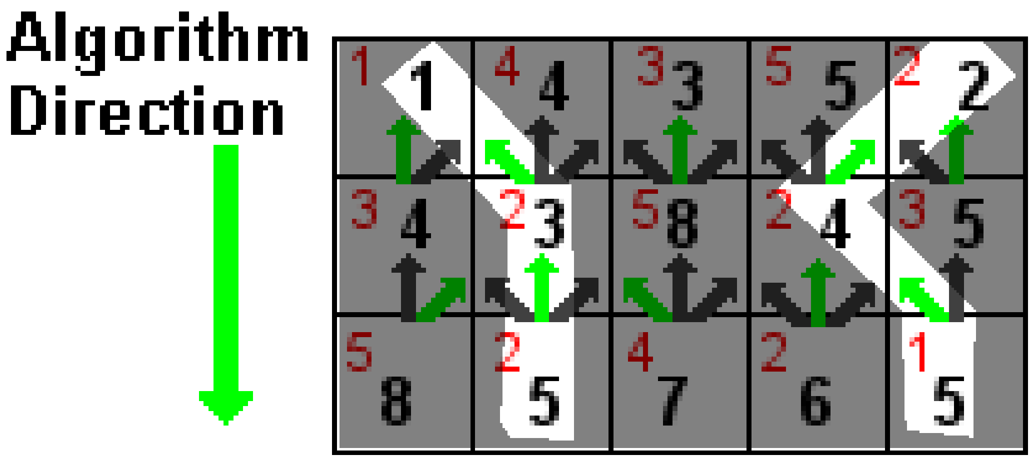


e = red numbe
M = black num
B = green arro



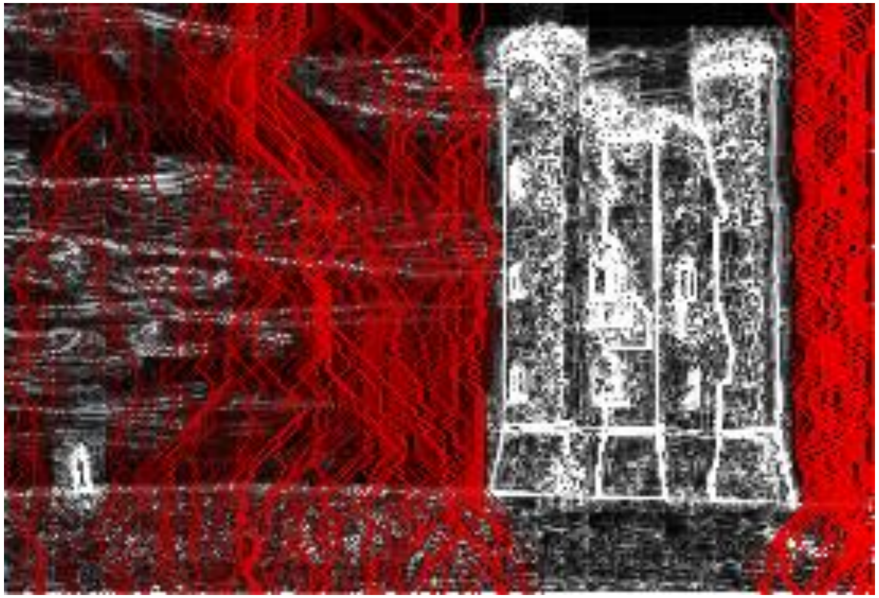
Credit: Wikipedia

Backward Pass



Credit: Wikipedia

Seams



Seams over energy image

Seams over input image

Shrink Image in 1 Dimension

- Change the image from size $n \times m$ to $n \times m'$
  - assume $m' < m$
- Remove $m\text{-}m' = c$ seams successively

Seam Carving

Shrink Image in 1 Dimension

- Change the image from size $n \times m$ to $n \times m'$
  - assume $m' < m$
- Remove $m\text{-}m' = c$ seams successively

Scaling

Shrink Image in Both Dimensions: Optimal Seam Ordering

- Change the image from size $n \times m$ to $n' \times m'$
  - assume $m' < m$ and $n' < n$
- What is the best order for seam carving?
  - Remove vertical seams first?
  - Horizontal seams first?
  - Alternate between the two?

Optimal Seam Ordering

- Solve optimization problem:

$$\min_{\mathbf{s^x},\mathbf{s^y},\alpha} \sum_{i=1}^{k} E(\alpha_i \mathbf{s_i^x} + (1-\alpha_i)\mathbf{s_i^y})$$

where $k = r+c$, $r = (m-m')$, $c = (n-n')$ and α_i is a parameter that determines if at step i we remove a horizontal or vertical seam: $\alpha \in \{0,1\}$

Enlarging Images

- Method 1: Compute the optimal vertical (horizontal) seam *s* in image and duplicate the pixels in *s* by averaging them with their left and right neighbors (top and bottom in the horizontal case)
- Often will choose the *same* seam at each iteration, producing noticeable stretching artifact

Enlarging Images

- Method 2: To enlarge width by *k*, compute top *k* vertical seams (for removal) and duplicate *each* of them

Content Amplification

- Scale the image; this will scale everything, “content” as well as “non-content”
- Shrink the scaled-image using seam carving, which will (hopefully) carve out the non-content part

Object Removal

- User marks the target object to be removed
- Force seams to pass through marked pixels
- Seams are removed from the image until all marked pixels are gone
- To obtain the original image size, use seam insertion

Object Removal

- One shoe removed (and image enlarged to original size)

input

Object Removal

- Object marking to prevent unwanted results: mark regions where seams must ***not*** pass

Multi-Size Images

- Methods mentioned so far are not real-time
- We calculate best seam, remove it, calculate the next seam based on new image, etc.
- For real-time resizing
  - Pre-compute removal order for every pixel in image
  - Compute Index map, V, of size $n \times m$ that encodes, for each pixel, the index of the seam that removed it, i.e., $V(i, j) = t$ means pixel (i, j) was removed by the t^{th} seam removal iteration

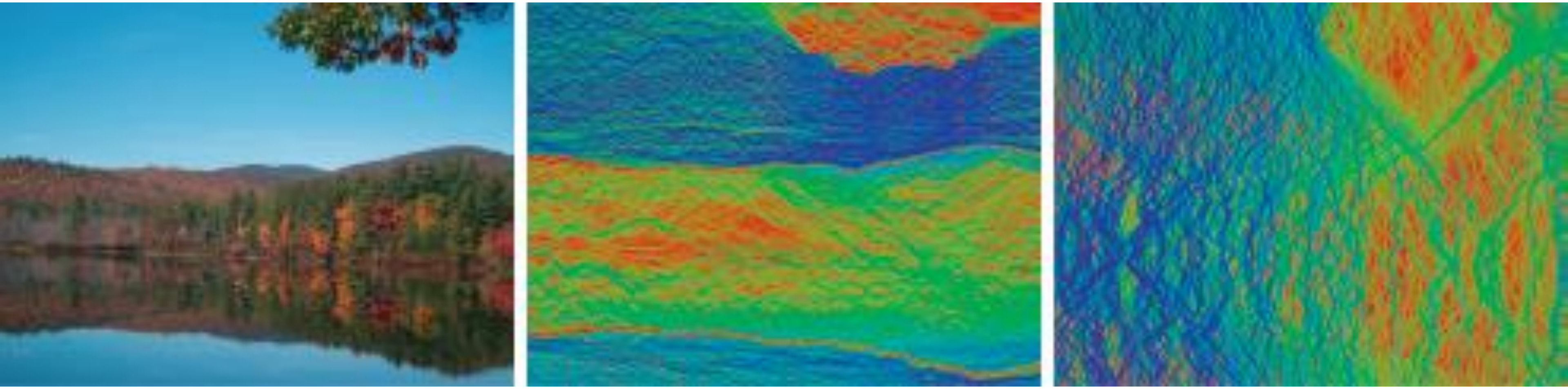
Multi-Size Images

- Horizontal Index map (H)
- Vertical Index map (V)

- To get an image of width m'
  - need to remove m-m' pixels from each row
  - concatenate, in each row, all pixels with seam index greater than or equal to $m-m'$

- Same for changing the height

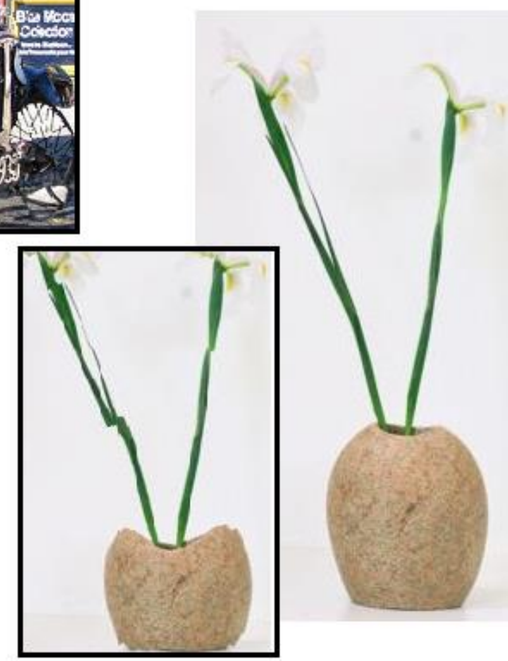
Seam Index Maps



Input H V

Blue seams are removed first, red seams removed last

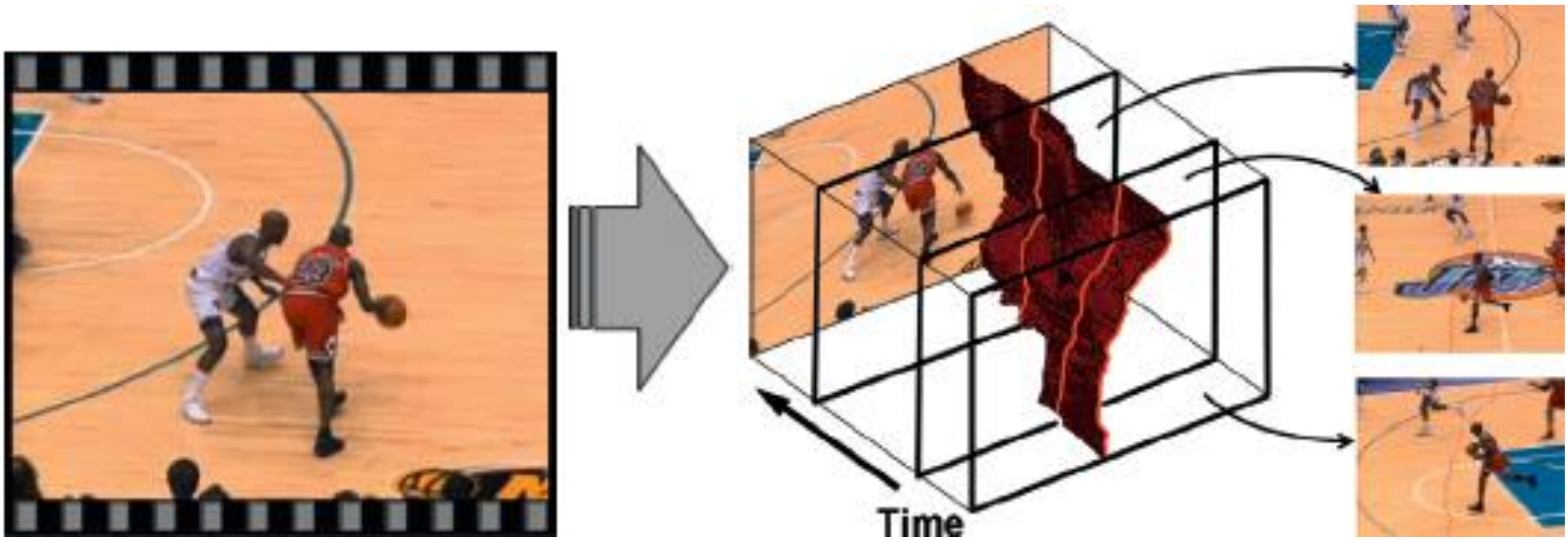
Failures



- Too much content
- No space for seam to avoid content

Video Resizing

- Resizing each frame independently is bad
- Instead, find 2D surface in 3D *x-y-t* video volume



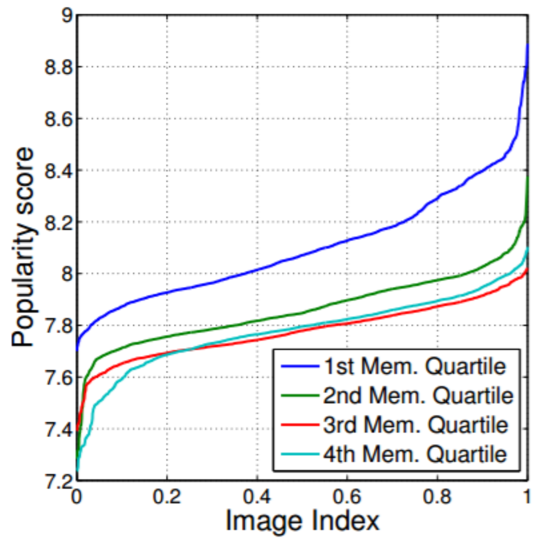
What next?

What makes a picture memorable?

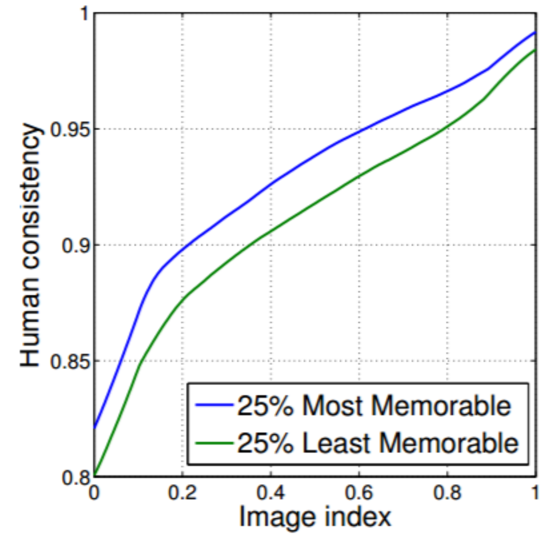
Most memorable images:
(top-left is most memorable…)

[Khosla et al. 2015]

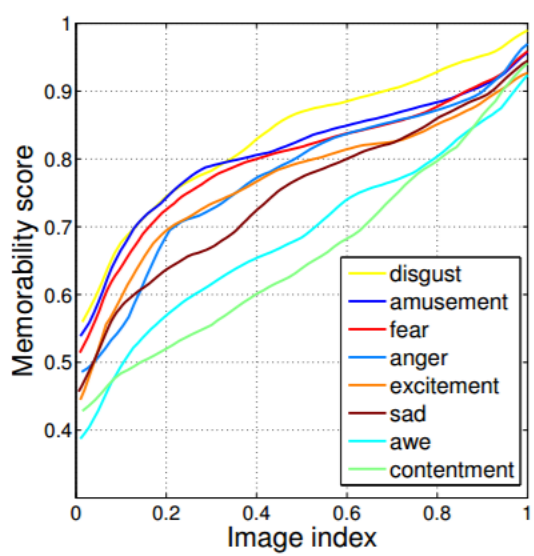
http://memorability.csail.mit.edu/



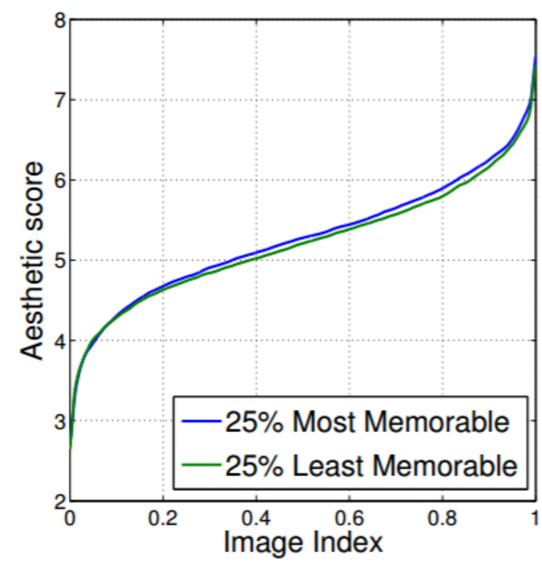
(a) Memorability vs popularity



(b) Memorability vs saliency



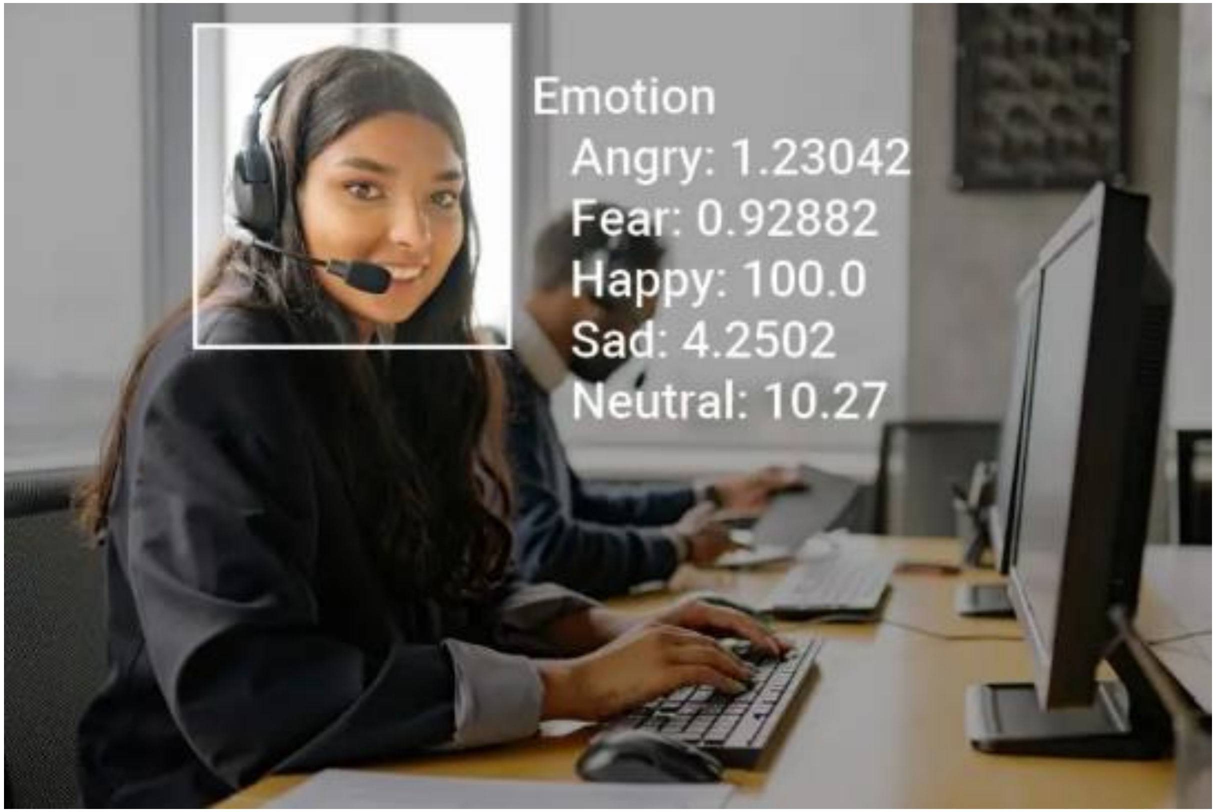
(c) Memorability vs emotions



(d) Memorability vs aesthetics

AI-Based Emotion Recognition

- https://viso.ai/deep-learning/visual-emotion-ai-recognition/
- https://openaccess.thecvf.com/content/CVPR2023W/LatinX/papers/de_Lima_Costa_High-Level_Context_Representation_for_Emotion_Recognition_in_Images_CVPRW_2023_paper.pdf



EmoGen: Emotional Image Content **Generation** with Text-to-Image Diffusion Models

- https://openaccess.thecvf.com/content/CVPR2024/papers/Yang_EmoGen_Emotional_Image_Content_Generation_with_Text-to-Image_Diffusion_Models_CVPR_2024_paper.pdf

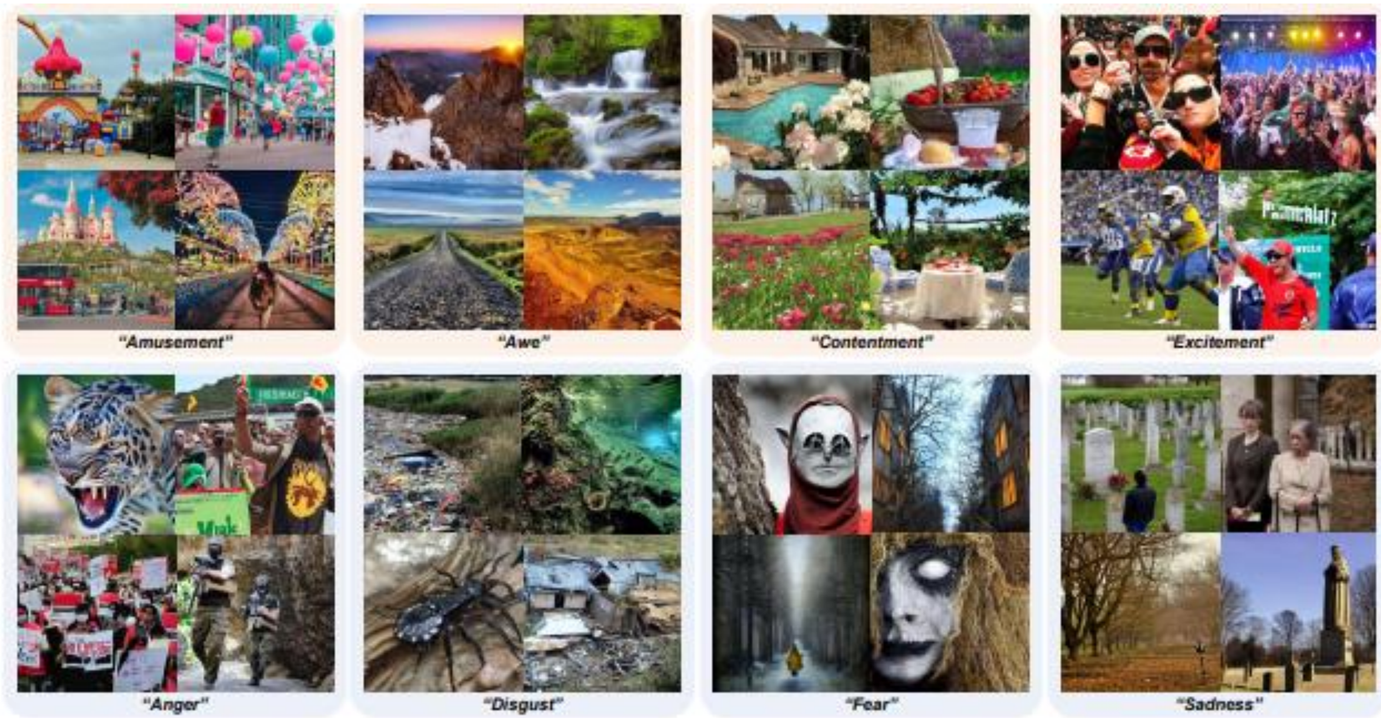


Figure 1. Emotional Image Content Generation (EICG). Given an emotion category, our network produces images that exhibit unambiguous meanings (*semantic-clear*), reflect the intended emotion (*emotion-faithful*) and incorporate varied semantics (*semantic-diverse*).

and More…

Mood Determination: urgency, emergency?