Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing

Image Warping and Morphing

* Warping and morphing applied to images

Georgia Tech College of Computing

Lesson Objectives

1. Image Warping
2. Forward and Inverse Warping
3. Warping using a mesh
4. Image Morphing
5. Feature-based Image Morphing

Recall: Image Transformations

image filtering:
change range of image

$$g(x) = T(f(x))$$

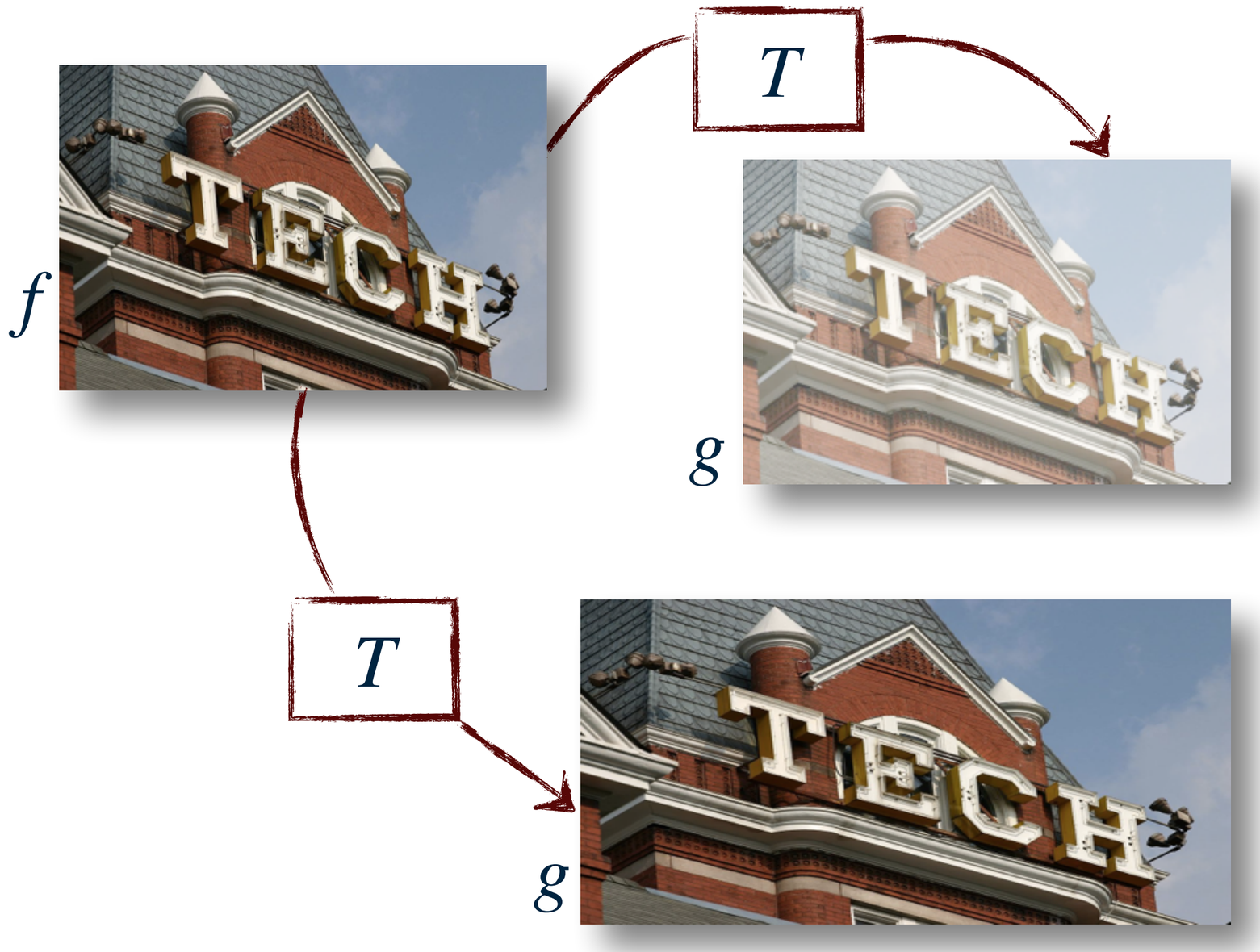


image warping:
change domain of image

$$g(x) = f(T(x))$$

Image Transformation vs. Warping

* Transformation: Lines remain lines
* Warping: Points are mapped to points
* A mathematical function for warping from a plane to the plane

Image Warping

- Distorted through simulation of optical aberrations
- Projected onto a curved or mirrored surface
- Partitioned into polygons and each polygon distorted
- Distorted using morphing

Two methods: Forward/Inverse

* Consider a S and T image
* S has pixel coordinates: (u,v)
* T has pixel coordinates: (x,y)

$S \Rightarrow (u,v)$

Forward $(x, y) = [X(u, v), Y(u, v)]$

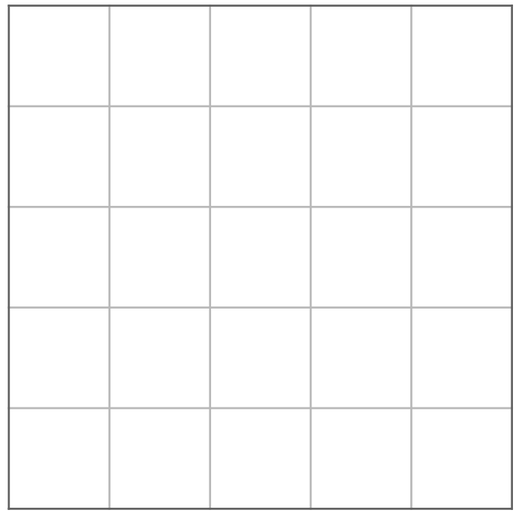
Inverse $(u, v) = [U(x, y), V(x, y)]$

$T \Rightarrow (x,y)$

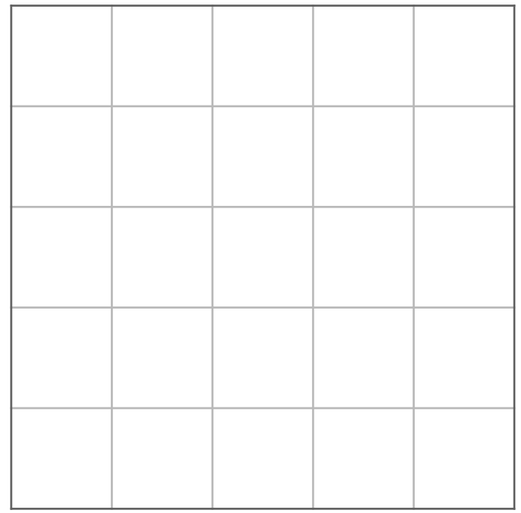
Forward/Inverse Warping

$$(x, y) = [X(u, v), Y(u, v)]$$

Input



(u,v) integer valued

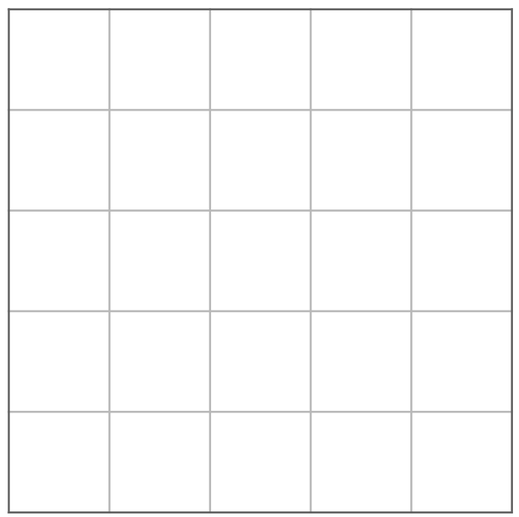


(x,y) real valued

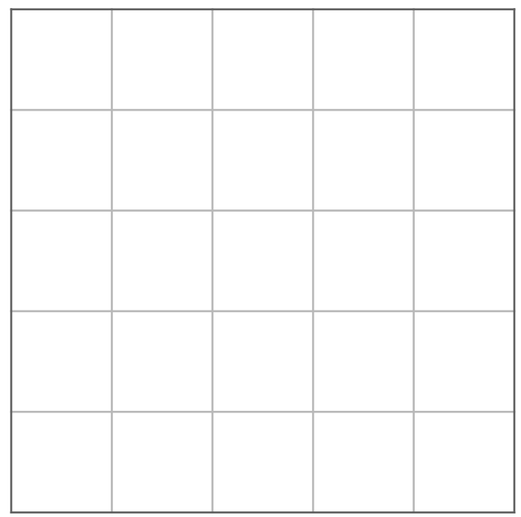
Output

$$(u, v) = [U(x, y), V(x, y)]$$

Input



(u,v) real valued



(x,y) integer valued

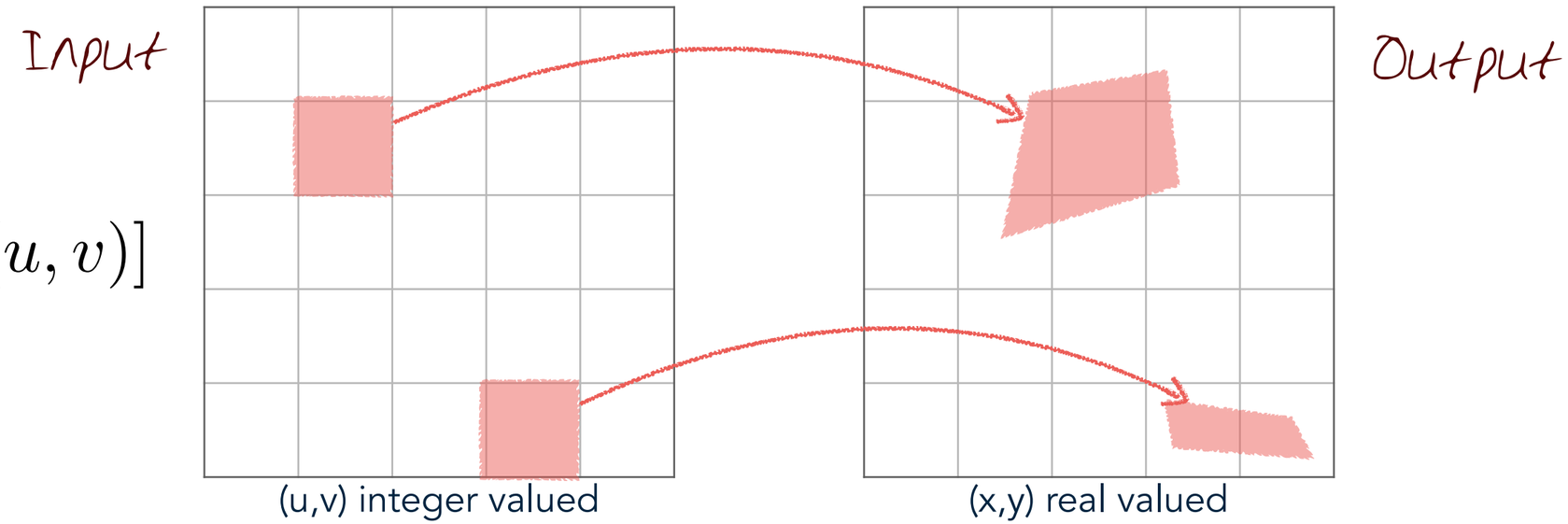
Output

Forward/Inverse Warping

$$(x, y) = [X(u, v), Y(u, v)]$$

Problems:

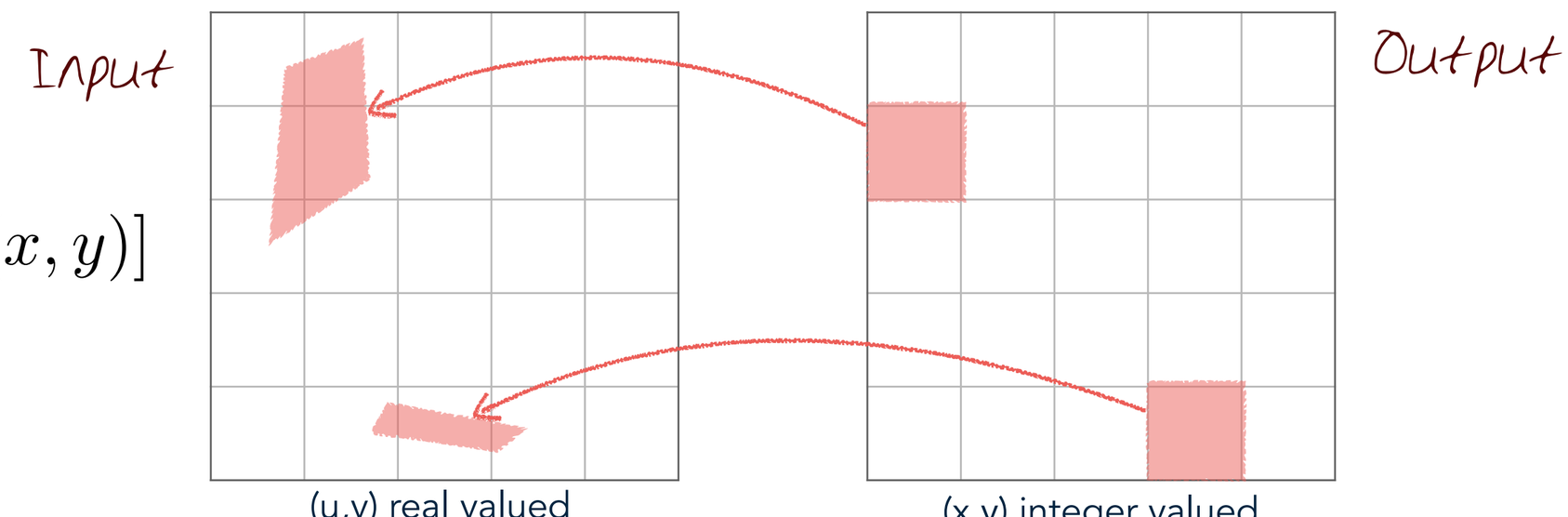
Holes, Overlaps



$$(u, v) = [U(x, y), V(x, y)]$$

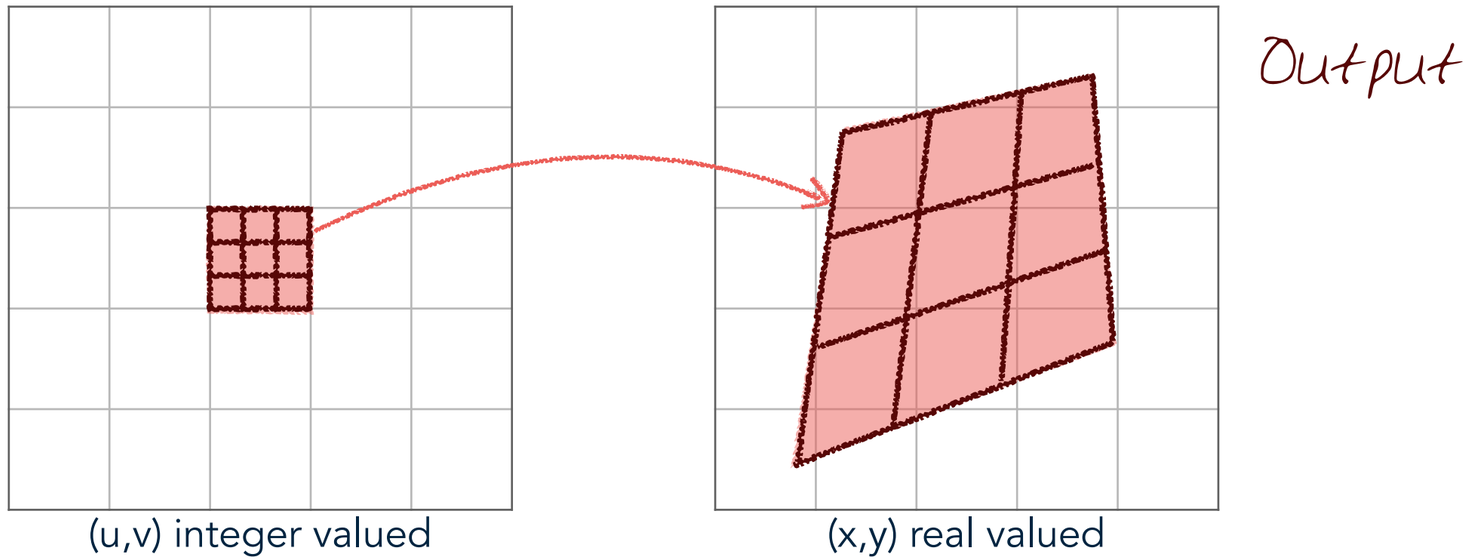
Problems:

Minification

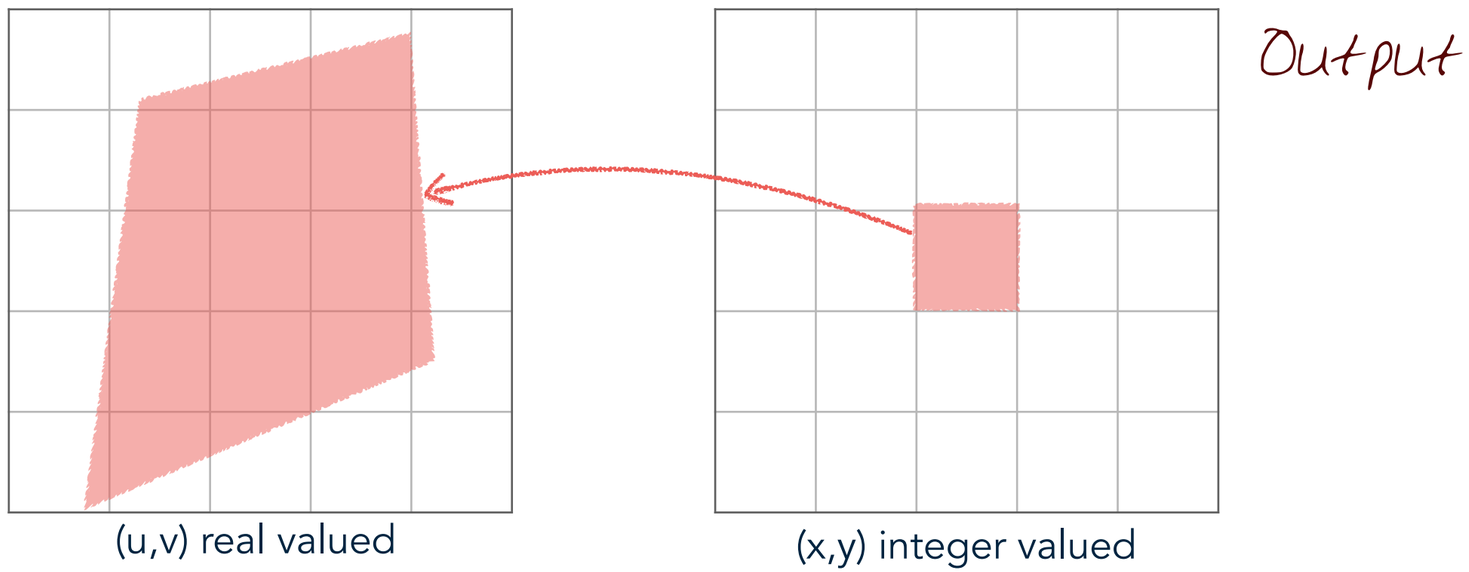


Magnification / Minification

$$(x, y) = [X(u, v), Y(u, v)]$$



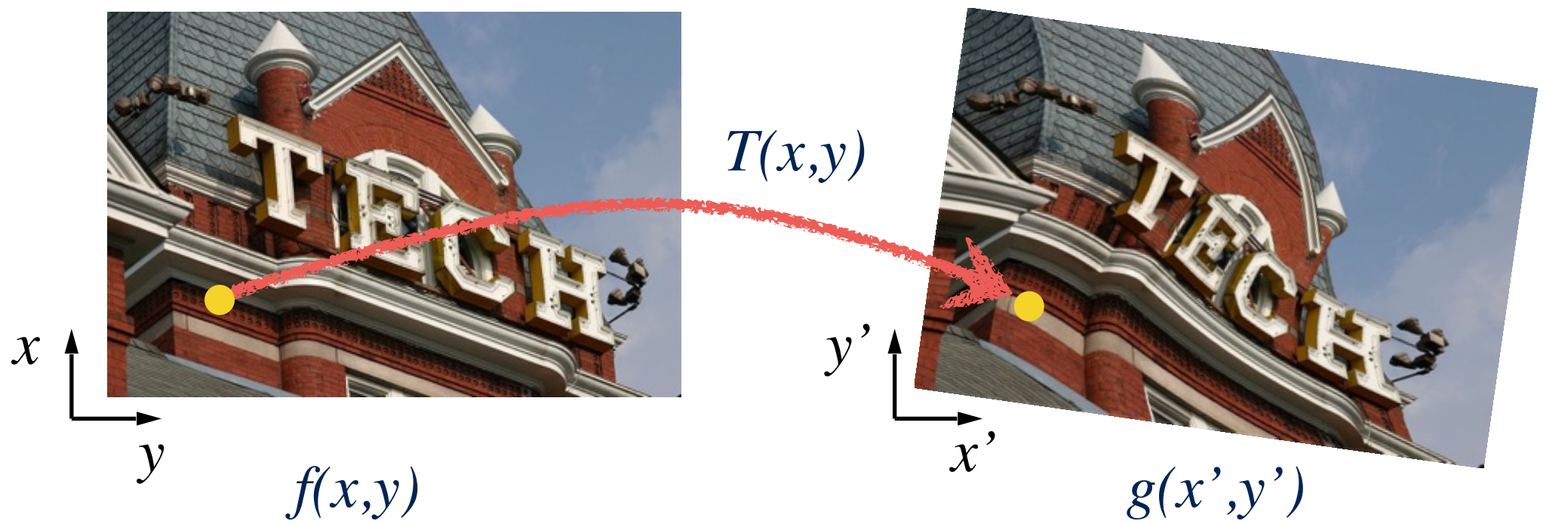
$$(u, v) = [U(x, y), V(x, y)]$$



Additional Info:

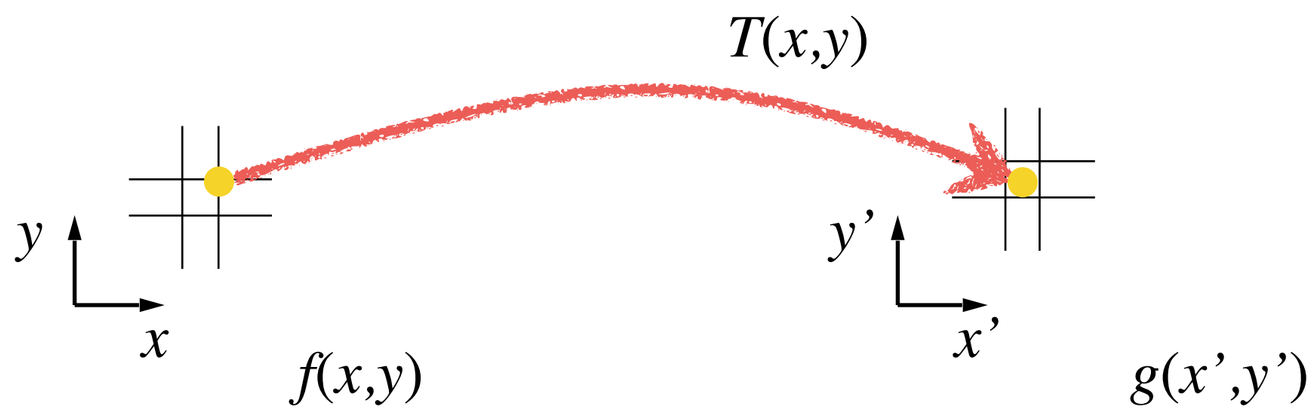
See Two-pass Transforms

Recall: Forward warping



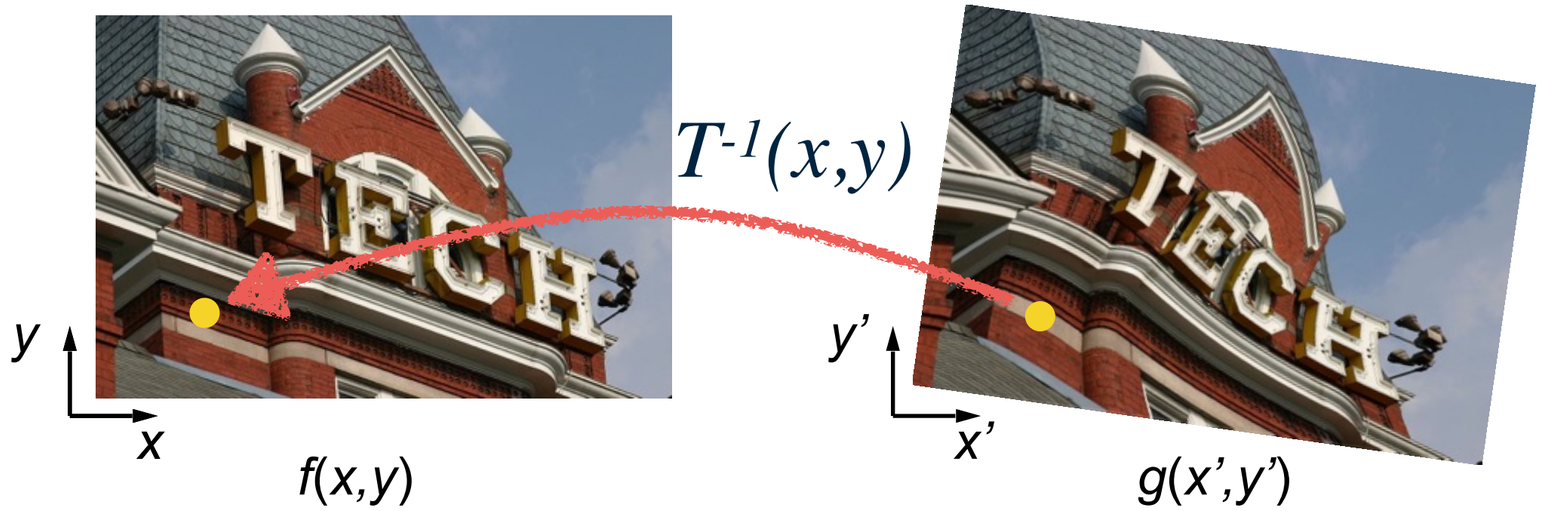
* Send each pixel $f(x,y)$ to its corresponding location
  * $(x',y') = T(x,y)$ in the second image
* Q: what if pixel lands "between" two pixels?

Recall: Forward warping



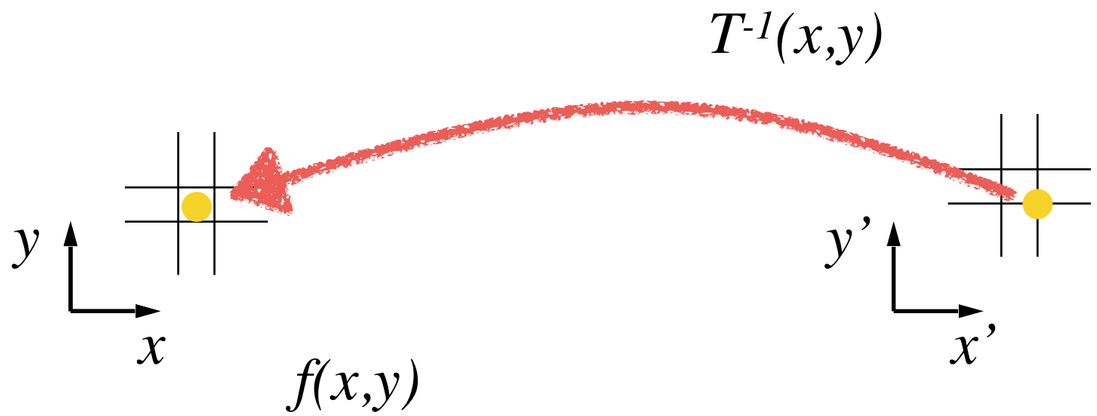
- Send each pixel $f(x,y)$ to its corresponding location
  - $(x',y') = T(x,y)$ in the second image
- Q: what if pixel lands "between" two pixels?
- A: distribute color among neighboring pixels (x',y')

Recall: Inverse warping



* Get each pixel $g(x',y')$ from its corresponding location
  * $(x,y) = T^{-1}(x',y')$ in the first image
* Q: what if pixel comes from "between" two pixels?

Recall: Inverse warping



* Get each pixel $g(x',y')$ from its corresponding location
  * $(x,y) = T^{-1}(x',y')$ in the first image
* Q: what if pixel comes from "between" two pixels?
* A: Interpolate color value from neighbors

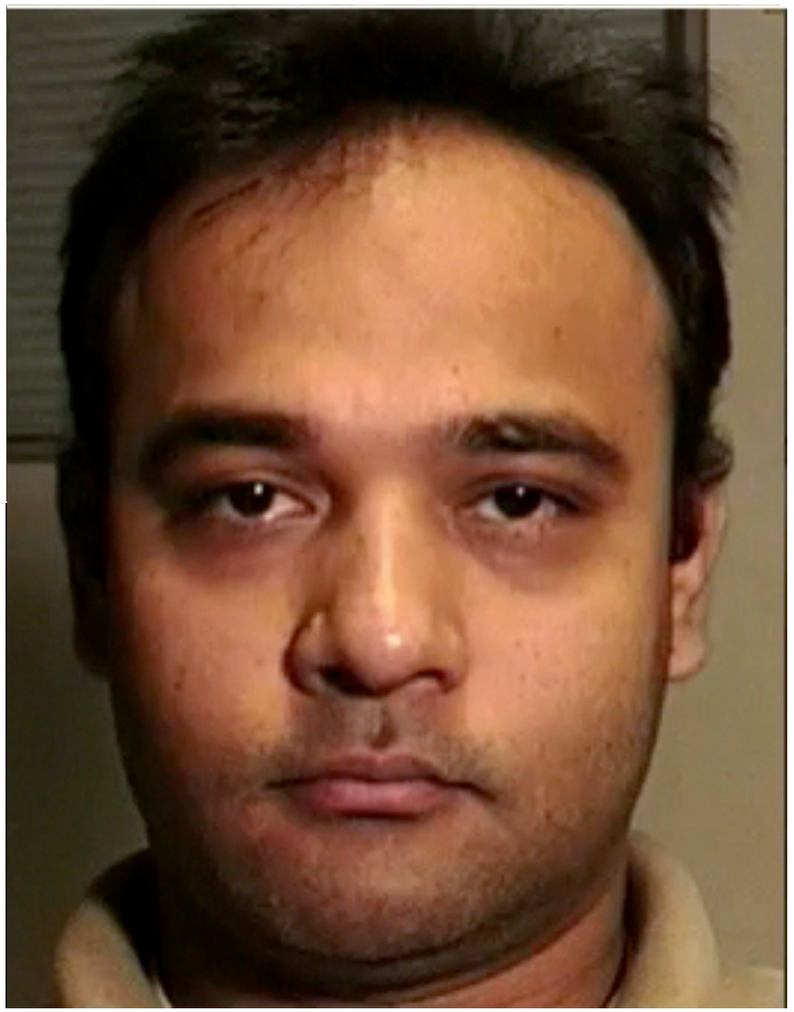
Forward vs. inverse warping

* Q: which is better?

* A: usually inverse ⇒ eliminates holes

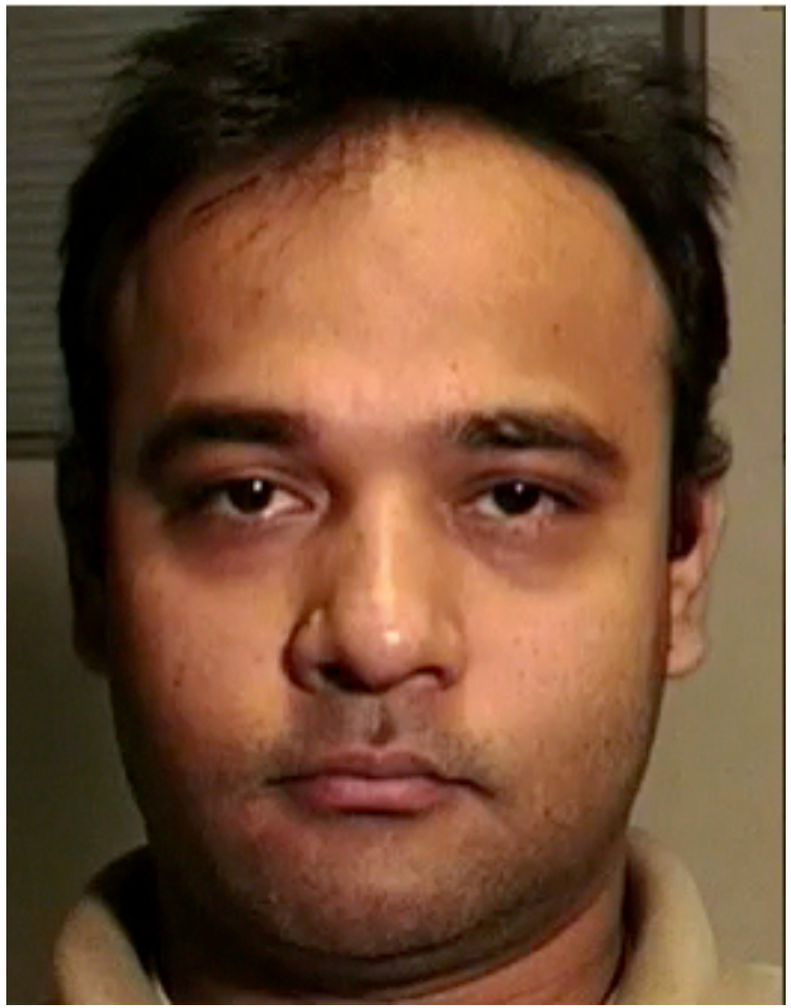
  * however, it requires an invertible warp function ⇒ not always possible...

Mesh-based Warping



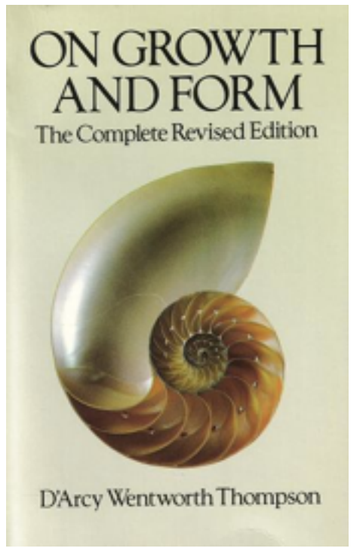
* Use a sparse set of corresponding points and interpolate with a displacement field
* Triangulate the set of points on Source
* Use the affine model for each triangle
* Triangulate Target with displaced Points
* Use inverse mapping

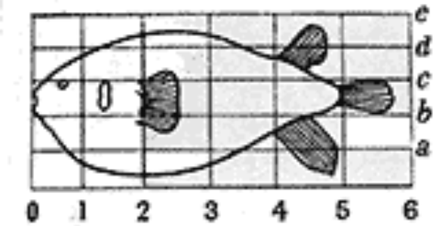
Image Morphing



* Animations that changes (or morphs) one image or shape into another through a seamless transition

* Widely used in movies





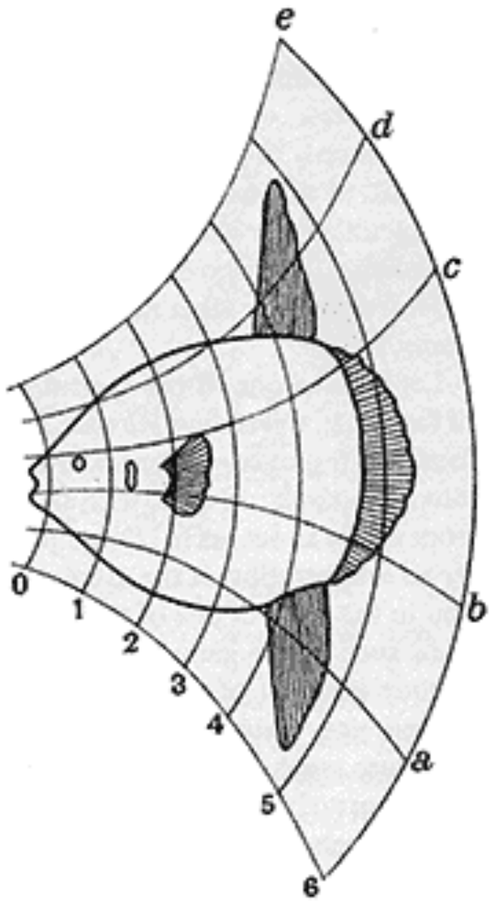
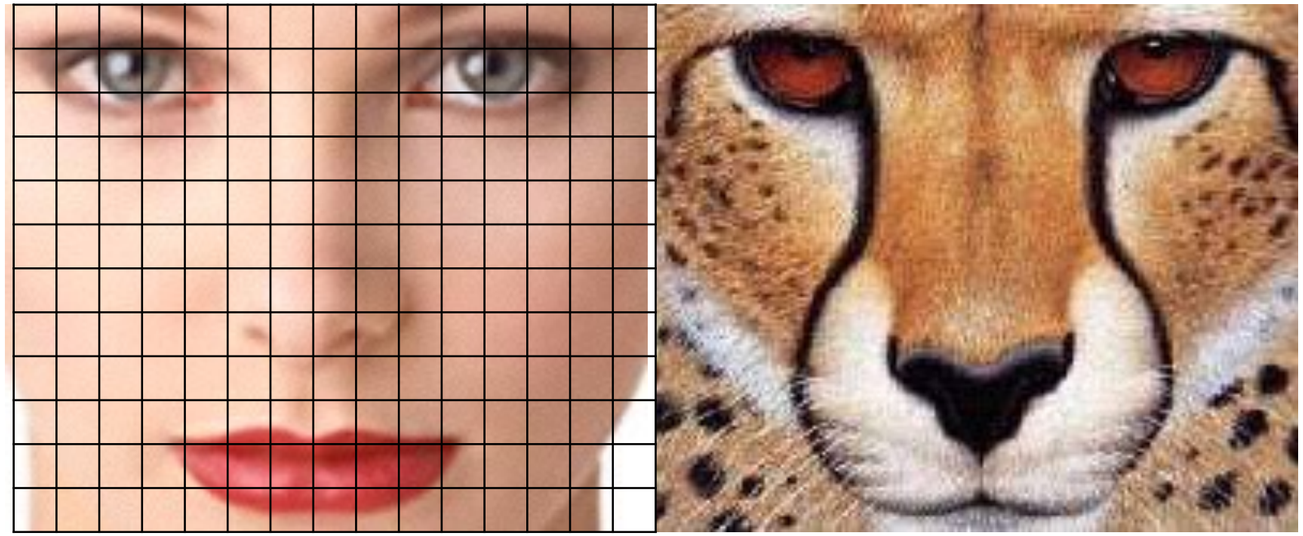


Image Morphing: Approaches

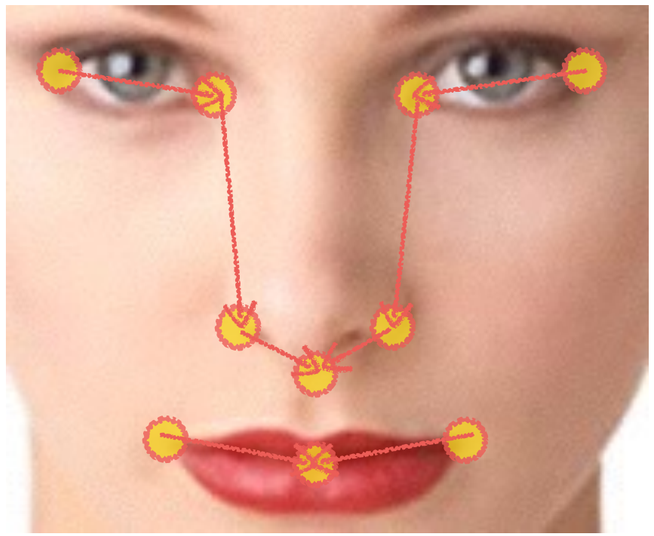
* Quadrilateral mesh displaced with variational interpolation

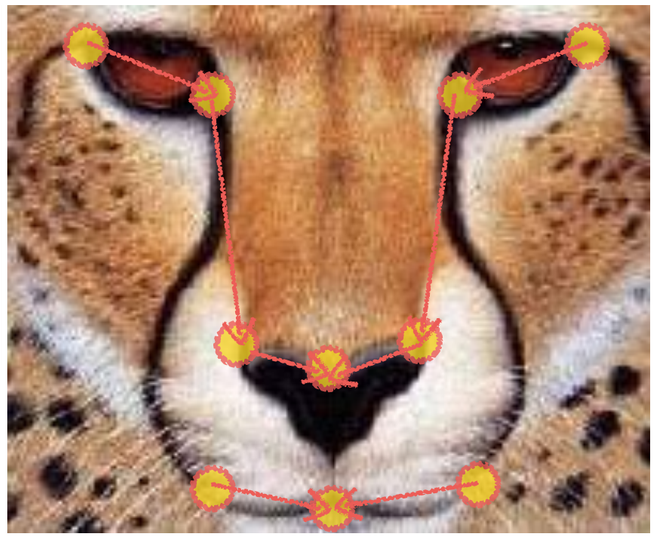


Images on this slide are for demonstration purposes only and may not reflect actual processed images
For more details see Szeliski Chapter 3

Image Morphing: Approaches

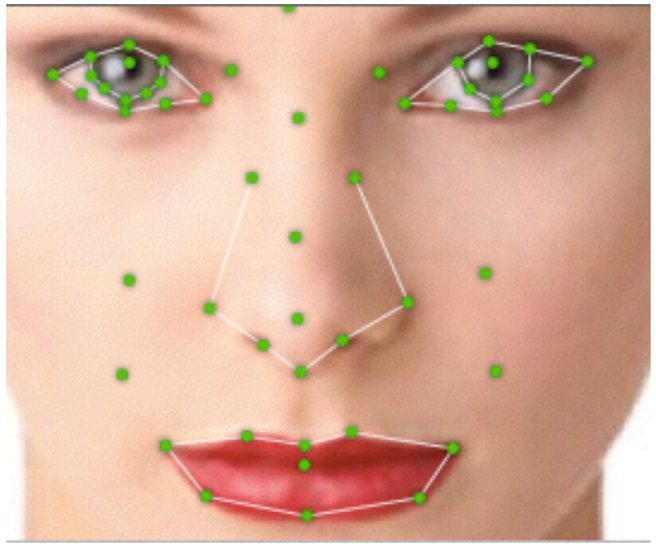
* Quadrilateral mesh displaced with variational interpolation
* Corresponding features/ points
* Corresponding orientated line segments (specifies translation, rotation, scaling)

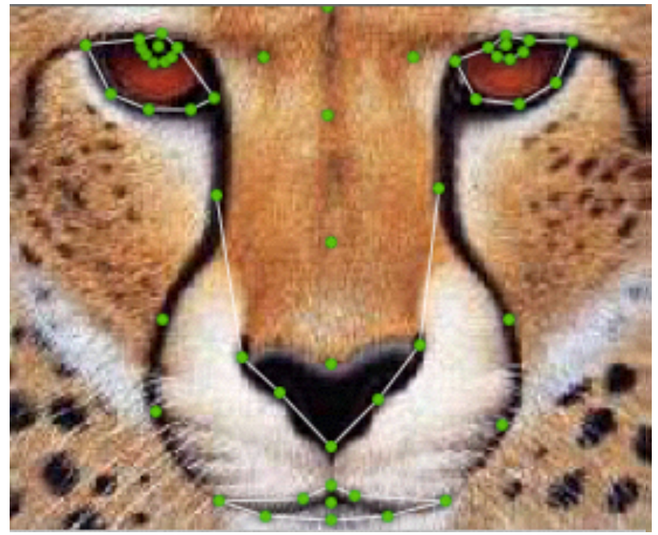




Images on this slide are for demonstration purposes only and may not reflect actual processed images
For more details see Szeliski Chapter 3

Feature-based Morphing





Beier and Neely (1992)

Software used: Fantamorph v5

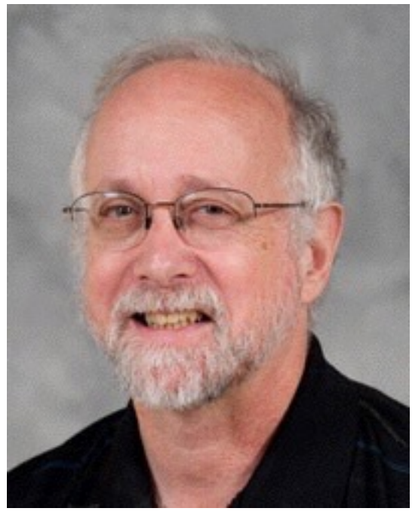
Feature-based Morphing

Michael Jackson - Black Or White

Beier and Neely (1992)

Feature-based Morphing

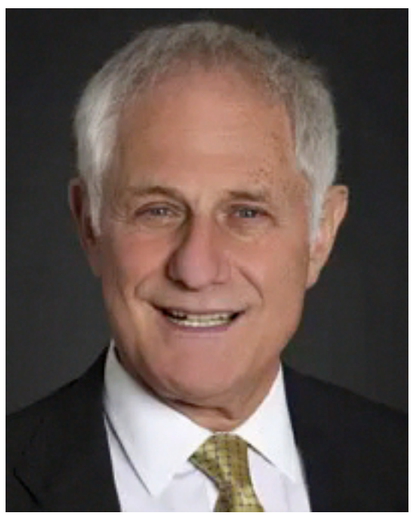
Zvi



Ron

Irfan

Charles



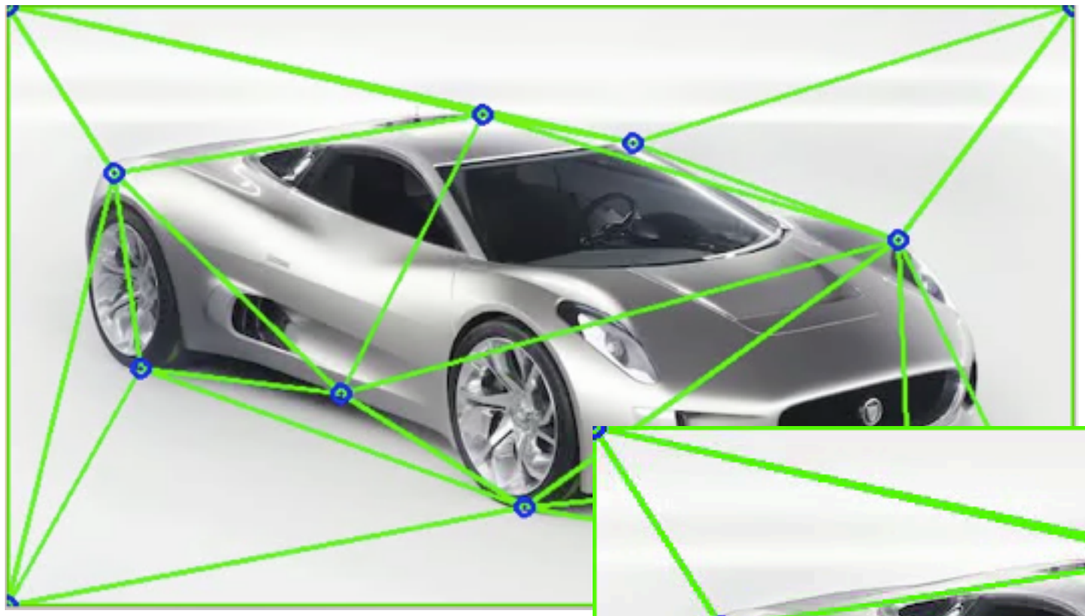
CoC Deans

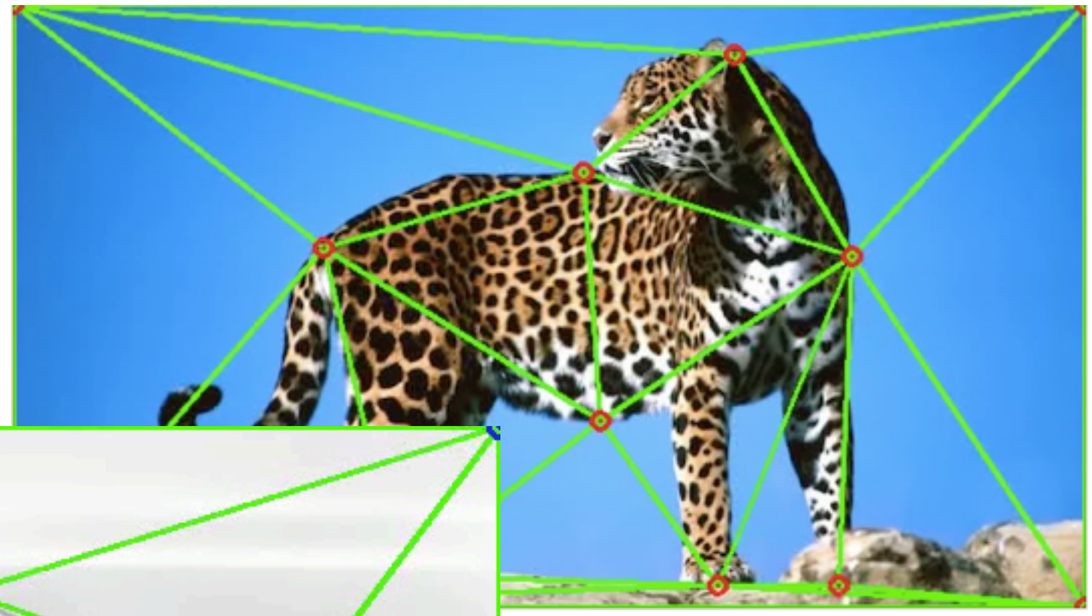
Morphed

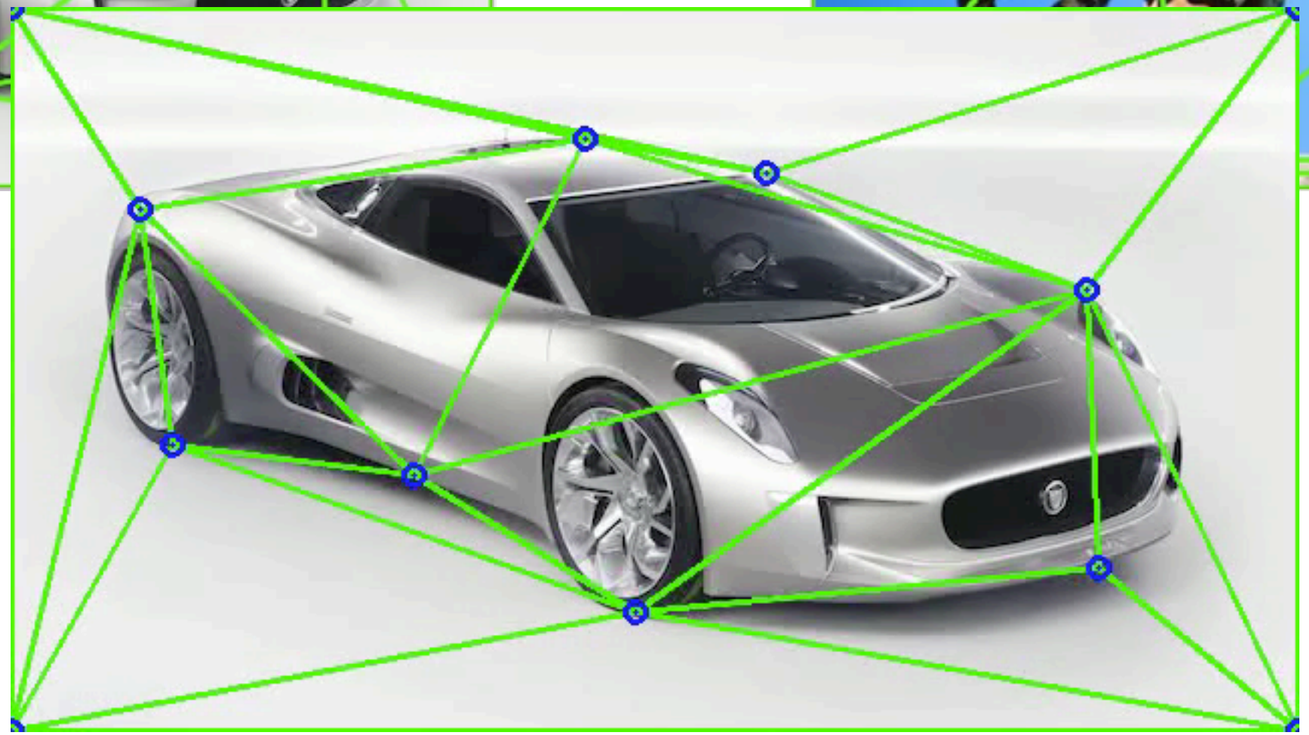
Beier and Neely (1992)

Software used: Fantamorph v5

Demo





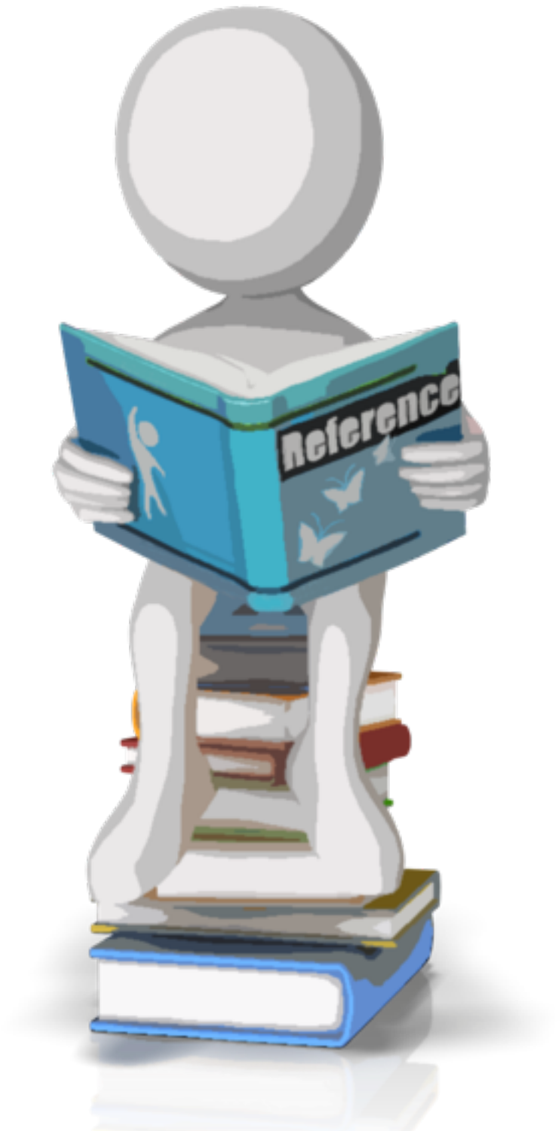


```
# Main script
if __name__ == "__main__":
    # Read two images: We want to morph from src to dst
    img_src = cv2.imread("jaguar-cx75.png")
    img_dst = cv2.imread("jaguar-standing.png")

    # Define corresponding point pairs (x, y)
    pts_src = np.float32(
        [[64, 100], [80, 217], [285, 65], [200, 232],
         [375, 82], [310, 300], [534, 140],[540, 278]])
    pts_dst = np.float32(
        [[185, 145], [230, 340], [340, 100], [350, 248],
         [430, 30], [420, 346], [500, 150], [492, 346]])

    # Visualize points [debug]
    '''
    img_src_out = img_src.copy()
    img_dst_out = img_dst.copy()
    drawPoints(img_src_out, pts_src, (255, 0, 0), (255, 0, 0))
    drawPoints(img_dst_out, pts_dst, (0, 0, 255), (0, 0, 255))
    cv2.imshow("src", img_src_out)
    cv2.imshow("dst", img_dst_out)
    '''

    # Call morphing function with src, dst images and points, and optional parameters
    num_frames = 31
    morph(img_src, img_dst, pts_src, pts_dst, num_frames, save_video=True, show_points=False,
show_triangles=False)
```

Summary

* More details about Image Warping
* Basics of Forward and Inverse Warping
* How to warp an image using a mesh
* Image Morphing
* Feature-based Image Morphing

Further Reading



* Beier and Neely (1992) "Feature-based Image Metamorphosis" ACM SIGGRAPH 1992

* Szeliski (2010) Chapter 3

Credits

* For more information, see:
  * Richard Szeliski (2010) Computer Vision: Algorithms and Applications, Springer (Chapter 3)
* Some concepts in slides motivated by similar slides by James Hays Alyosha Efros and Greg Turk
* Additional list will be available on website

Computational Photography

* Study the basics of computation and its impact on the entire workflow of photography, from capturing, manipulating and collaborating on, and sharing photographs.

Georgia Tech College of Computing