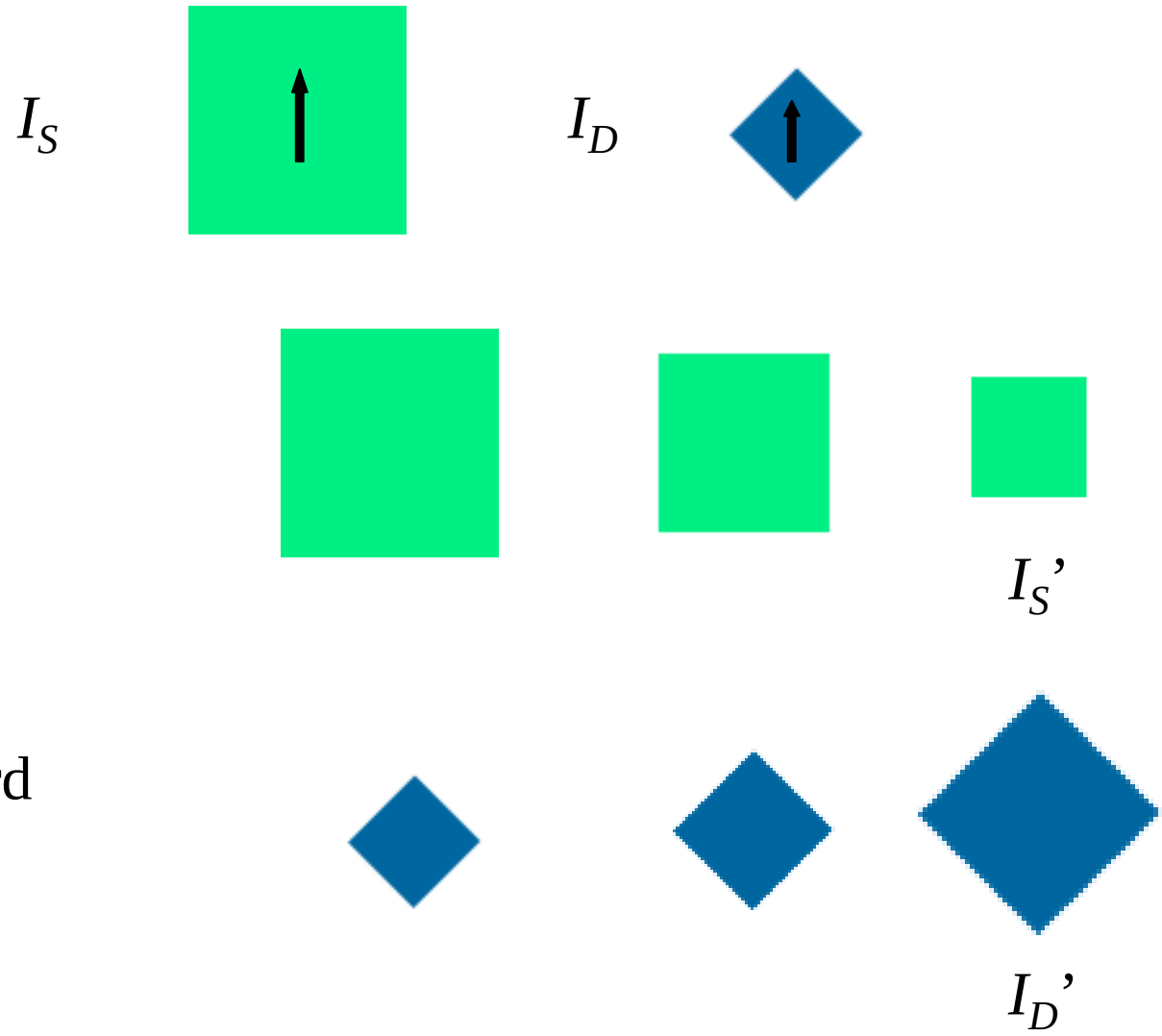
Computer Science 631
Lecture 2: Morphing

Ramin Zabih
Computer Science Department
CORNELL UNIVERSITY

How does morphing work?

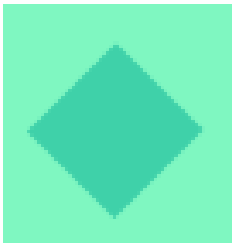
- Overview
  - Draw a ray on corresponding feature on I_S and I_D
  - Make a video of I_S changing its shape to be like I_D (align the rays)
  - Make a video of I_D changing its shape be be like I_S (align the rays)
  - Play the first video forwards, the second video backwards, and blend them
    - Compute a (pixel-by-pixel) weighted average

Example



Resulting video

- Blend the two videos (dissolve)
  - Initially, 100% forward video
  - At the end, 100% backward video
  - In between, you can control the speed of the morph



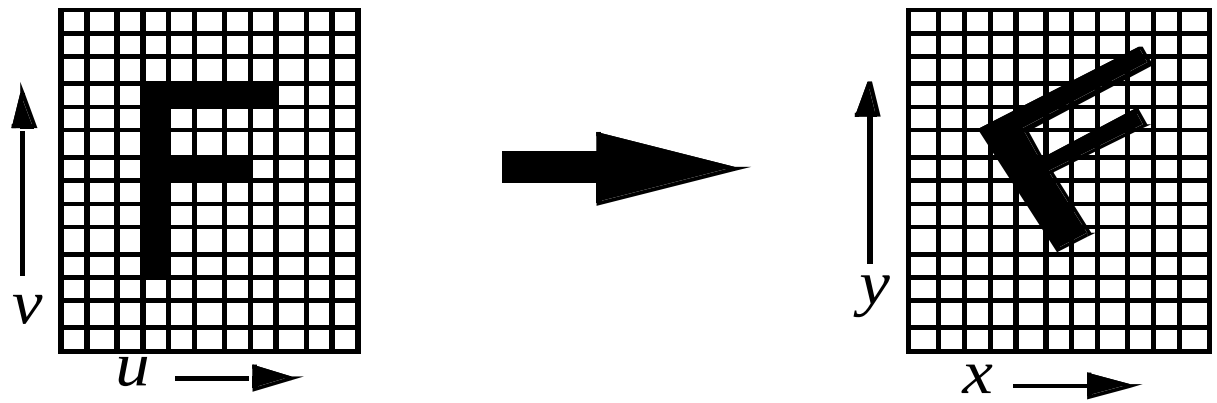
average of half-shrunk green and half-grown blue

Warping

- We need to change the “shape” of the input image I_S in a well-specified manner
  - This is done by drawing a ray (oriented line segment) on I_S and a corresponding ray on I_D
    - Sometimes called “control lines”
  - The way this feature changes between I_S and I_D controls the change in the shape of I_S

Coordinate change

- We need a function that turns points on the input image *(u,v)* into points on the output image *(x,y)*



Source scanning

For each pixel at coordinates *(u,v)*

```
x = x(u,v);
y = y(u,v);
dest[x,y] = source[u,v];
```

- One can also do this from destination to source (“destination scanning”)

What should our functions x,y be?

- If $x(u,v) = u+2$ and $y(u,v) = v+3$, what do we do to the image?
  - Translate by 2 left and 3 down?
  - 2 right and 3 up?
- We need at least allow translations and rotations

Affine transformations

$x = au + bv + c$
$y = du + ev + f$

- This gives a combination of rotation, scaling and translation
  - $a = e = 1$, $b = d = 0$ gives a translation by (c,f)
  - $a = e = cos\ \theta$, b = sin θ, d = - sin θ, c = f = 0 gives a rotation by θ
  - $a = s$, $e = s$, others 0 gives a scaling by s

Affine transformation properties

- An affine transformation will map a square to a parallelogram
  - More precisely, for every square and parallelogram, there is a corresponding affine transformation (and vice-versa)
- There are 6 degrees of freedom

Projective transformations

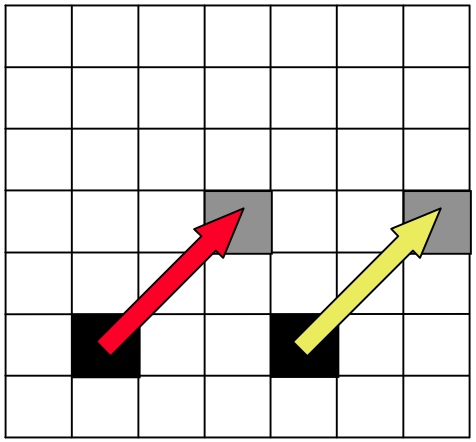
$$x = (au + bv + c)/(gu + hv + 1)$$
$$y = (du + ey + f)/ (gu + hv + 1)$$

- Affine is obviously a special case
- Allows us to map any square into any quadrilateral
- 8 degrees of freedom

Morphing transformations

- The transformations used in morphing are different
  - More intuitive, easier to control
- Every ray defines a local coordinate system, so the motion of a ray (from source to destination) defines a transformation

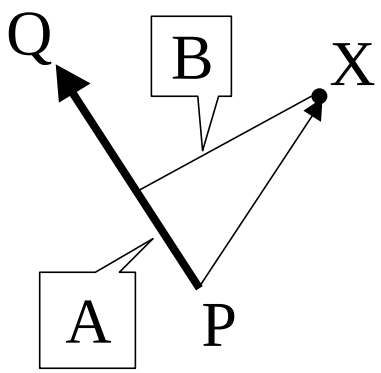
Interpolate the motion of the rays



- Red arrow in I_S goes to yellow arrow in I_D
  - Easy to create two intermediate rays
  - In general, we interpolate the motion of the endpoints of the ray

Ray-based coordinates

- Compute the position of the pixel X w.r.t. an oriented ray PQ
  - Coordinates are A (along PQ) and B (perpendicular to PQ)



$$A = \frac{(Q-P)}{\|Q-P\|} \cdot (X-P)$$

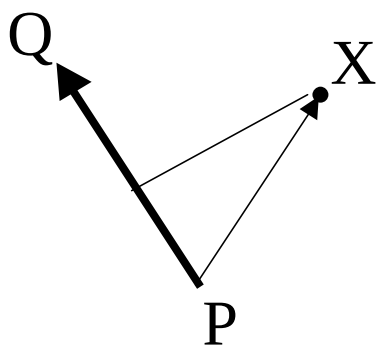
$$B = \frac{(Q-P)^{\perp}}{\|Q-P\|} \cdot (X-P)$$

Changing units

- The problem is that A and B are in units of pixels
  - Need them in percentages of the length of PQ

$$u = \frac{A}{\|Q-P\|} = \frac{(Q-P)}{\|Q-P\|^2} \cdot (X-P)$$

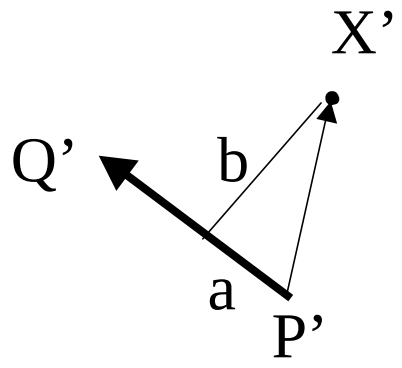
$$v = \frac{B}{\|Q-P\|} = \frac{(Q-P)^{\perp}}{\|Q-P\|^2} \cdot (X-P)$$

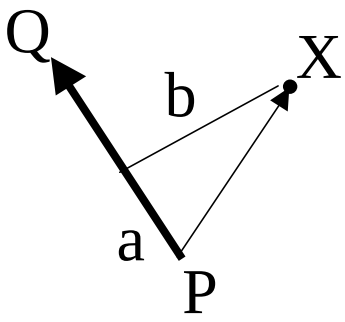


Putting it all together

- Want to warp I_S to create *I2*
  - The ray P'Q' moved to the ray PQ
  - X is the pixel in *I2* whose intensity we want
- Compute position of X w.r.t. PQ
  - The intensity should come from the point in that position with respect to P'Q' in I_S

General formula





$$X' = P' + a \cdot (Q' - P') + b \cdot (Q' - P')^{\perp}$$

Note that X’ (as well as X) is a *point*, not a *pixel*