Lecture 8

Image Transformations

(global and local warps)

Handouts: PS#2 assigned

Last Time

Idea #1: Cross-Dissolving / Cross-fading

- Interpolate whole images:

  $I_{halfway} = \alpha * I_1 + (1 - \alpha) * I_2$

- This is called **cross-dissolving** in film industry
- But what if the images are not aligned?

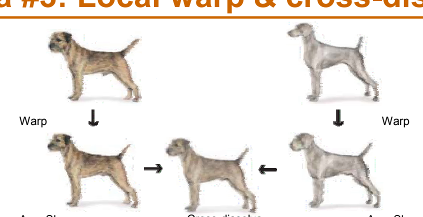
Idea #2: Align, then cross-disolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Failure: Global warping

- What to do?
  - Cross-dissolve doesn't work
  - Global alignment doesn't work
    - Cannot be done with a global transformation (e.g. affine)
  - Any ideas?
- Feature matching!
  - Nose to nose, tail to tail, etc.
  - This is a local (non-parametric) warp

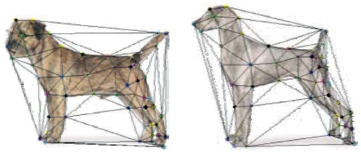
Idea #3: Local warp & cross-dissolve



Morphing procedure:

1. Find the average shape (the "mean dog"☺)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

Triangular Mesh



1. Input correspondences at key feature points
2. Define a triangular mesh over the points
   (ex. Delaunay Triangulation)
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately
   - How do we warp a triangle?
   - ***3 points = affine warp!***
   - Just like texture mapping

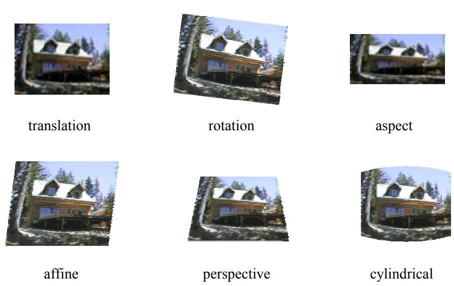
Slide Alyosha Efros

Transformations

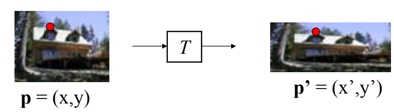
(global and local warps)

Parametric (global) warping

- Examples of parametric warps:



Parametric (global) warping



- Transformation T is a coordinate-changing machine:

  p' = *T*(**p**)

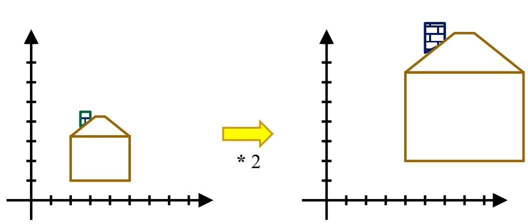
- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let's represent *T* as a matrix:

  p' = Tp

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

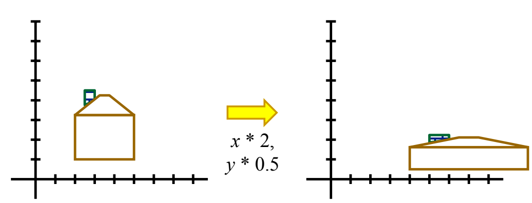
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



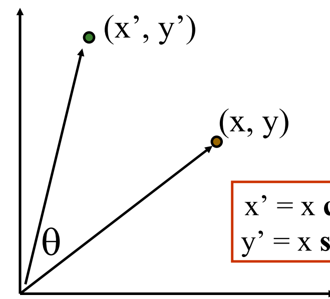
Scaling

- Scaling operation: $x' = ax$
  $y' = by$
- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } \mathbf{S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

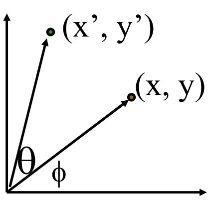
What's inverse of **S**?

2-D Rotation



$$\boxed{\begin{aligned} x' &= x\ \mathbf{cos}(\theta) - y\ \mathbf{sin}(\theta) \\ y' &= x\ \mathbf{sin}(\theta) + y\ \mathbf{cos}(\theta) \end{aligned}}$$

2-D Rotation



x = r cos (ϕ)
y = r sin (ϕ)
x' = r cos (ϕ + θ)
y' = r sin (ϕ + θ)

Trig Identity…
x' = r (cos(ϕ) cos(θ) – sin(ϕ) sin(θ))
y' = r (sin(ϕ) cos(θ) + cos(ϕ) sin(θ))

Substitute…
x' = x **cos**(θ) - y **sin**(θ)
y' = x **sin**(θ) + y **cos**(θ)

2-D Rotation

- This is easy to capture in matrix form:

x' = x **cos**(θ) - y **sin**(θ)
y' = x **sin**(θ) + y **cos**(θ) →

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Even though sin(θ) and cos(θ) are nonlinear functions of θ,
  - ***x' is a linear combination of x and y***
  - ***y' is a linear combination of x and y***
- What is the inverse transformation?
  - Rotation by – θ
  - For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Identity?

$$x' = x \qquad y' = y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Scale around (0,0)?

$$\boldsymbol{x' = s_x * x} \qquad \boldsymbol{y' = s_y * y} \qquad \begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 \\ 0 & \boldsymbol{s_y} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Rotate around (0,0)?

$$x' = \cos\Theta * x - \sin\Theta * y \qquad y' = \sin\Theta * x + \cos\Theta * y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Shear?

$$\boldsymbol{x' = x + sh_x * y} \qquad \boldsymbol{y' = sh_y * x + y} \qquad \begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} \\ \boldsymbol{sh_y} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

$$x' = -x \qquad y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror over (0,0)?

$$x' = -x \qquad y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x \qquad y' = y + t_y$$

NO!

Only linear 2D transformations can be represented with a 2x2 matrix

All 2D Linear Transformations

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror
- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x \qquad y' = y + t_y$$

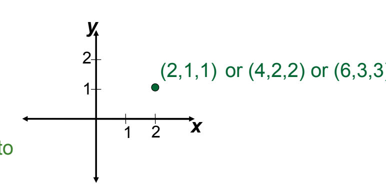
Homogeneous Coordinates

- ***Homogeneous coordinates***
  - represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneou s coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (x, y, 0) represents a point at infinity
  - (0, 0, 0) is not allowed



Convenient coordinate system to represent many useful transformations

Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x$$
$$y' = y + t_y$$

- A: Using the rightmost column:

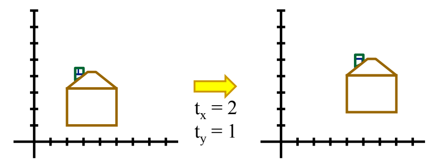
$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

- Example of translation

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Matrix Composition

- Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

p' = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ **p**

Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

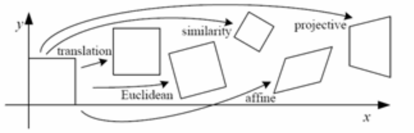
- Affine transformations are combinations of
  - Linear transformations, and
  - Translations
- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition
  - Models change of basis
- Will the last coordinate *w* always be 1?

Projective Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Models change of basis

2D image transformations

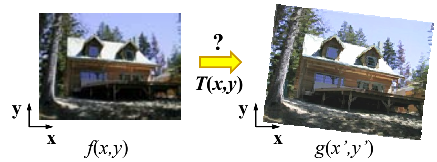


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$			
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$			
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$			
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$			
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$			

These transformations are a nested set of groups

- Closed under composition and inverse is a member

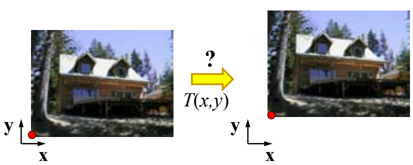
Recovering Transformations



- What if we know *f* and *g* and want to recover the transform T?
  - willing to let user provide correspondences
    - How many do we need?

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

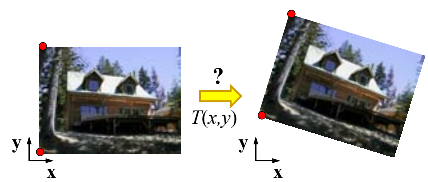
Translation: # correspondences?



- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

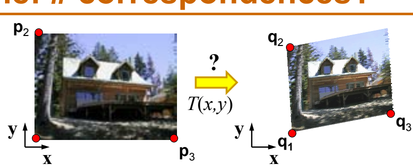
Euclidian: # correspondences?



- How many correspondences needed for translation + rotation?
- How many DOF?

$$T = s\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine: # correspondences?

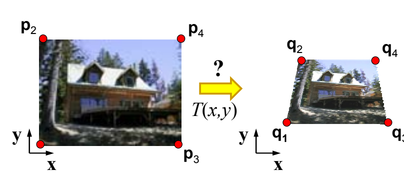


- How many correspondences needed for affine?
- How many DOF?
- An affine transformation is a composition of translations, rotations, dilations, and shears.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & k_x & 0 \\ k_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

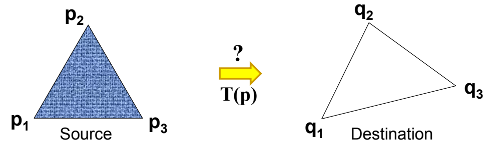
translations rotation dilations shear

Projective: # correspondences?



- How many correspondences needed for projective?
- How many DOF?

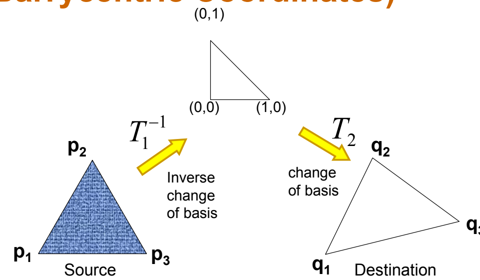
Warping triangles



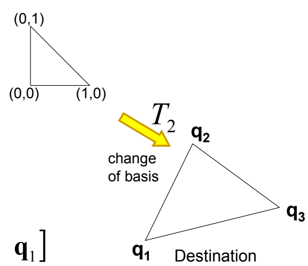
- Given two triangles: $p_1p_2p_3$ and $q_1'q_2'q_3'$ in 2D (6 constraints)
- Need to find transform **T** to transfer all pixels from one to the other.
- What kind of transformation is **T**?
  - affine
- How can we compute the transformation matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Hint: Warping triangles (Barycentric Coordinates)

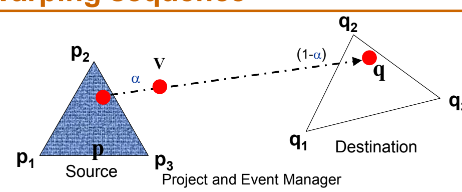


Trick: Computing T



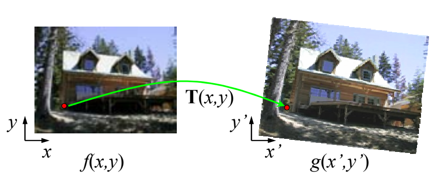
$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{q}_2 - \mathbf{q}_1 & \mathbf{q}_3 - \mathbf{q}_1 & \mathbf{q}_1 \end{bmatrix}$$

Warping sequence



$$\mathbf{v} = \mathbf{p} + \alpha(t)(\mathbf{q} - \mathbf{p})$$

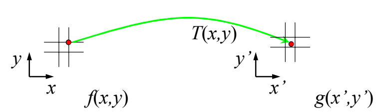
Forward warping



- Send each pixel $f(x,y)$ to its corresponding location $(x',y') = \mathbf{T}(x,y)$ in the second image

Q: what if pixel lands between four pixels?

Forward warping

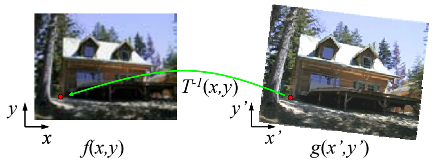


- Send each pixel $f(x,y)$ to its corresponding location $(x',y') = \mathbf{T}(x,y)$ in the second image

Q: what if pixel lands between four pixels?

A: distribute color among neighboring pixels (x',y')
- Known as "splatting"
- Check out `griddata` in Matlab

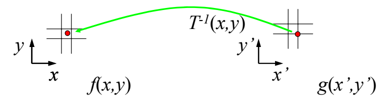
Inverse warping



- Get each pixel $g(x',y')$ from its corresponding location
  $(x,y) = \mathbf{T}^{-1}(x',y')$ in the first image

Q: what if pixel comes from between four pixels?

Inverse warping



- Get each pixel $g(x',y')$ from its corresponding location
  $(x,y) = \mathbf{T}^{-1}(x',y')$ in the first image

Q: what if pixel comes from between four pixels?

A: *Interpolate* color value from neighbors
  - nearest neighbor, bilinear, Gaussian, bicubic
  - Check out `interp2` in Matlab

Forward vs. inverse warping

- Q: which is better?
- A: usually inverse—eliminates holes
  - however, it requires an invertible warp function—not always possible...