Matching Planar Objects In New Viewpoints … And Much More
– via Homography
What Transformation Happened To My DVD?

- Rectangle goes to a parallelogram
Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

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

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
- Rectangles go to parallelograms

[Source: N. Snavely, slide credit: R. Urtasun]
What Transformation Really Happened To My DVD?

- What about now?
What Transformation Really Happened To My DVD?

- Actually a rectangle goes to **quadrilateral**
2D Image Transformations

- These transformations are a nested set of groups
- Closed under composition and inverse is a member

[source: R. Szeliski]
Projective Transformations

- Homography:

\[
\begin{bmatrix}
w \\
wz \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}
\]

Properties:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are \textbf{not} preserved
- Closed under composition
- Rectangle goes to quadrilateral
- Affine transformation is a special case, where \( g = h = 0 \) and \( i = 1 \)

[Source: N. Snavely, slide credit: R. Urtasun]
What Transformation Really Happened to My DVD?

For **planar** objects:

- Viewpoint change for planar objects is a **homography**
- Affine transformation **approximates** viewpoint change for planar objects that are far away from camera
What Transformation Happened to My DVD?

- Why should I care about homography?
- Now that I care, how should I estimate it?
- I want to understand the geometry behind homography. That is, why aren’t parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?
Homography

- Why should I care about homography? Let’s answer this first
- Now that I care, how should I estimate it?
- I want to understand the geometry behind homography. That is, why aren’t parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?
Why do we need homography? Can’t we just assume that the transformation is affine? The approximation on the right looks pretty decent to me...

That’s right. If I want to detect (match) an object in a new viewpoint, an affine transformation is a relatively decent approximation.
Homography

Why do we need homography? Can’t we just assume that the transformation is affine? The approximation on the right looks pretty decent to me...

That’s right. If I want to detect (match) an object in a new viewpoint, an affine transformation is a relatively decent approximation.

But for some applications I want to be more accurate. Which?
Why do we need homography? Can’t we just assume that the transformation is affine? The approximation on the right looks pretty decent to me...

That’s right. If I want to detect (match) an object in a new viewpoint, an affine transformation is a relatively decent approximation.

But for some applications I want to be more accurate. Which?
Application 1: a Little Bit of CSI

- Tom Cruise is taking an exam on Monday
The professor keeps the exams in this office
He enters (without permission) and takes a picture of the laptop screen
His picture turns out to not be from a viewpoint he was shooting for (it’s difficult to take pictures while hanging)

Can he still read the exam?
Warping an Image with a Global Transformation

Transformation $T$ is a coordinate-changing machine:

$$[x', y'] = T(x, y)$$

What does it mean that $T$ is global?

- Is the same for any point $p$
- Can be described by just a few numbers (parameters)

[Source: N. Snavely, slide credit: R. Urtasun]
Warping an Image with a Global Transformation

Example of warping for different transformations:
Forward and Inverse Warping

- **Forward Warping:** Send each pixel $f(x)$ to its corresponding location $(x', y') = T(x, y)$ in $g(x', y')$

  ```plaintext
  procedure forwardWarp(f, h, out g):
  For every pixel $x$ in $f(x)$
  1. Compute the destination location $x' = h(x)$.
  2. Copy the pixel $f(x)$ to $g(x')$.
  ```

- **Inverse Warping:** Each pixel at destination is sampled from original image

  ```plaintext
  procedure inverseWarp(f, h, out g):
  For every pixel $x'$ in $g(x')$
  1. Compute the source location $x = \hat{h}(x')$
  2. Resample $f(x)$ at location $x$ and copy to $g(x')$
  ```

[source: R. Urtasun]
Application 1: a Little Bit of CSI

- We want to transform the picture (plane) inside these 4 points into a rectangle (laptop screen)
We want it to look like this. How can we do this?
A transformation that maps a projective plane (a quadrilateral) to another projective plane (another quadrilateral, in this case a rectangle) is a homography.
Application 1: a Little Bit of CSI

If we compute the homography and warp the image according to it, we get this

Pick up the books and stop cheating!

Go home and learn about:
1. Scale invariant interest points
2. SIFT descriptor
3. Matching via homography
4. Robust matching
Application 1: a Little Bit of CSI

If we used affine transformation instead, we’d get this. Would be even worse if our picture was taken closer to the laptop.
Application 1: a Little More of CSI

What is the shape of the b/w floor pattern?

The floor (enlarged)

Automatically rectified floor

Slide from Antonio Criminisi
Application 1: a Little More of CSI

Slide from Antonio Criminisi

From Martin Kemp *The Science of Art*  
(manual reconstruction)
Application 1: a Little More of CSI

What is the (complicated) shape of the floor pattern?

Automatically rectified floor

St. Lucy Altarpiece, D. Veneziano
Slide from Criminisi
Application 1: a Little More of CSI

From Martin Kemp, *The Science of Art*
*(manual reconstruction)*

Automatic rectification

Slide from Criminisi
Application 2: How Much do Soccer Players Run?
How many meters did this player run?
Field is planar. We know its dimensions (look on Wikipedia).
Application 2: How Much do Soccer Players Run?

- Let’s take the 4 corner points of the field
Application 2: How Much do Soccer Players Run?

- We need to compute a homography that maps them to these 4 corners.
We need to compute a homography that maps the 4 corners. Any other point from this plane (the field) also maps to the right with the same homography.
Nice. What happened to the players?
We can now also transform the player’s trajectory and we have it in meters!
Application 2: How Much do Soccer Players Run?

- If we used affine transformation... Our estimations of running would not be accurate!
Application 3: Panorama Stitching

Take a tripod, rotate camera and take pictures

[Source: Fernando Flores-Mangas]
Application 3: Panorama Stitching

[Source: Fernando Flores-Mangas]
Application 3: Panorama Stitching

Each pair of images is related by homography! **If we also moved the camera, this wouldn’t be true** (next class)

[Source: Fernando Flores-Mangas]
To do panorama stitching, we need to:

- Match points between pairs of images \( I \) and \( J \)
- Compute a transformation between the between matches in \( I \) and \( J \) : a homography
- Do it robustly (RANSAC)
- Warp the first image to the second using the estimated homography

Apart from the last point, this is exactly the same procedure as for the problem of matching planar objects across viewpoints

So this should motivate the why do I care part of the homographies
Homography

- Why should I care about homography?
- Now that I care, how should I estimate it? **Let’s do this now**
- I want to understand the geometry behind homography. That is, why aren’t parallel lines mapped to parallel lines in oblique viewpoints? How did we get that equation for computing the homography?
Solving for Homographies

- Let \((x_i, y_i)\) be a point on the reference (model) image, and \((x'_i, y'_i)\) its match in the test image.

- A homography \(H\) maps \((x_i, y_i)\) to \((x'_i, y'_i)\):

\[
\begin{bmatrix}
ax'_i \\
ay'_i \\
a
\end{bmatrix} =
\begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x_i \\
y_i \\
1
\end{bmatrix}
\]
Let \((x_i, y_i)\) be a point on the reference (model) image, and \((x'_i, y'_i)\) its match in the test image.

A homography \(H\) maps \((x_i, y_i)\) to \((x'_i, y'_i)\):

\[
\begin{bmatrix}
ax'_i \\
ay'_i \\
a
\end{bmatrix}
= \begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x_i \\
y_i \\
1
\end{bmatrix}
\]

We can get rid of that \(a\) on the left:

\[
x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]

\[
y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]
Solving for Homographies

- Let \((x_i, y_i)\) be a point on the reference (model) image, and \((x'_i, y'_i)\) its match in the test image.

- A homography \(H\) maps \((x_i, y_i)\) to \((x'_i, y'_i)\):

\[
\begin{bmatrix}
ax'_i \\
ya'_i \\
a
\end{bmatrix} = \begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix} \begin{bmatrix}
x_i \\
y_i \\
1
\end{bmatrix}
\]

- We can get rid of that \(a\) on the left:

\[
x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]

\[
y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]

- Hmmmm... Can I still rewrite this into a linear system in \(h\)?

[Source: R. Urtasun]
Solving for homographies

From:

\[
\begin{align*}
x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\
y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}
\end{align*}
\]

We can easily get this:

\[
\begin{align*}
x_i' (h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\
y_i' (h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12}
\end{align*}
\]

Rewriting it a little:

\[
\begin{align*}
h_{00}x_i + h_{01}y_i + h_{02} - x_i' (h_{20}x_i + h_{21}y_i + h_{22}) &= 0 \\
h_{10}x_i + h_{11}y_i + h_{12} - y_i' (h_{20}x_i + h_{21}y_i + h_{22}) &= 0
\end{align*}
\]
Solving for homographies

- We can re-write these equations:

\[
\begin{align*}
    h_{00}x_i + h_{01}y_i + h_{02} - x'_i (h_{20}x_i - h_{21}y_i - h_{22}) &= 0 \\
    h_{10}x_i + h_{11}y_i + h_{12} - y'_i (h_{20}x_i - h_{21}y_i - h_{22}) &= 0
\end{align*}
\]

- as a linear system!

\[
\begin{bmatrix}
    x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\
    0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \\
    \end{bmatrix}
\begin{bmatrix}
    h_{00} \\
    h_{01} \\
    h_{02} \\
    h_{10} \\
    h_{11} \\
    h_{12} \\
    h_{20} \\
    h_{21} \\
    h_{22} \\
\end{bmatrix} =
\begin{bmatrix}
    0 \\
    0 \\
\end{bmatrix}
\]

[Source: R. Urtasun]
Solving for homographies

- Taking all our matches into account:

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 & x'_1 & -x'_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 & x'_1 & -y'_1 \\
  x_n & y_n & 1 & 0 & 0 & 0 & -x'_n & x'_n & -x'_n \\
  0 & 0 & 0 & x_n & y_n & 1 & -y'_n & x'_n & -y'_n \\
\end{bmatrix}
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22} \\
\end{bmatrix} = \begin{bmatrix}
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{bmatrix}
\]

\[
A_{2n \times 9} h_{9 \times 2n} = 0
\]
Solving for homographies

- Taking all our matches into account:

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\
  x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\
  0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n
\end{bmatrix}
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22}
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

\[A_{2n \times 9} h_{9 \times 1} = 0_{9 \times 1}\]

- How many matches do I need to estimate \( H \)?
- This defines a least squares problem:

\[\min_{h} \|Ah\|_2^2\]
Solving for homographies

• Taking all our matches into account:

\[
\begin{bmatrix}
    x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
    0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
    x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
    0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \\
\end{bmatrix}
\begin{bmatrix}
    h_{00} \\
    h_{01} \\
    h_{02} \\
    h_{10} \\
    h_{11} \\
    h_{12} \\
    h_{20} \\
    h_{21} \\
    h_{22}
\end{bmatrix} = \begin{bmatrix}
    0 \\
    0 \\
    0 \\
    0 \\
\end{bmatrix}
\]

\[A_{2n \times 9} \begin{bmatrix} h \end{bmatrix}_9 = \begin{bmatrix} 0 \end{bmatrix}_{2n}\]

• How many matches do I need to estimate \( H \)?

• This defines a least squares problem:

\[
\min_h \|Ah\|_2^2
\]

• Since \( h \) is only defined up to scale, solve for unit vector
Solving for homographies

- Taking all our matches into account:

\[
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 & -y'_1 & -x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 & -y'_1 & -y'_1 \\
x_n & y_n & 1 & 0 & 0 & 0 & -x'_n & -y'_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -y'_n & -y'_n & -y'_n 
\end{bmatrix}
\begin{bmatrix}
h_{00} \\
h_{01} \\
h_{02} \\
h_{10} \\
h_{11} \\
h_{12} \\
h_{20} \\
h_{21} \\
h_{22} 
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 
\end{bmatrix}
\]

- How many matches do I need to estimate $H$?
- This defines a least squares problem:

\[
\min_{\mathbf{h}} \| \mathbf{Ah} \|_2^2
\]

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector
- Solution: $\hat{\mathbf{h}} = \text{eigenvector of } \mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
Solving for homographies

- Taking all our matches into account:

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
  0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
\end{bmatrix}
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22}
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0 \\
  \vdots \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

\[
A_{2n \times 9} h_{9} = 0_{2n}
\]

- How many matches do I need to estimate \( H \)?

- This defines a least squares problem:

\[
\min_{\mathbf{h}} \|A\mathbf{h}\|_2^2
\]

- Since \( \mathbf{h} \) is only defined up to scale, solve for unit vector

- Solution: \( \hat{\mathbf{h}} = \text{eigenvector of } A^T A \) with smallest eigenvalue

- Works with 4 or more points

[Source: R. Urtasun]
Image Alignment Algorithm: Homography

Given images $I$ and $J$

1. Compute image features for $I$ and $J$
2. Match features between $I$ and $J$
3. Compute **homography** transformation $A$ between $I$ and $J$ (with RANSAC)
Image Alignment Algorithm: Homography

Given images $I$ and $J$

1. Compute image features for $I$ and $J$
2. Match features between $I$ and $J$
3. Compute **homography** transformation $A$ between $I$ and $J$ (with RANSAC)

[Source: N. Snavely]
Panorama Stitching: Example 1

- Compute the matches

[Source: R. Queiroz Feitosa]
Panorama Stitching: Example 1

- Estimate the homography and warp

[Source: R. Queiroz Feitosa]
Panorama Stitching: Example 1

- Stitch

[Source: R. Queiroz Feitosa]
Panorama Stitching: Example 2

[Source: Fernando Flores-Mangas]
Panorama Stitching: Example 2

[Source: Fernando Flores-Mangas]
Panorama Stitching: Example 2

Laplacian Pyramid Blending ↓ seams not visible anymore

(Brown & Lowe; ICCV 2003)  google "Lowe Brown Autostitch"

[Source: Fernando Flores-Mangas]
A homography is a mapping between projective planes
You need at least 4 correspondences (matches) to compute it

Matlab functions:

- **TFORM = MAKETFORM(’AFFINE’,[X1,Y1],[X2,Y2]);** % Computes affine transformation between points [x1, y1] and [x2, y2]. Needs 3 pairs of matches (x1, y1, x2, y2 have three rows)
- **TFORM = MAKETFORM(’PROJECTIVE’,[X1,Y1],[X2,Y2]);** % Computes homography between points [x1, y1] and [x2, y2]. Needs 4 pairs of matches
- **IMW = IMTRANSFORM(IM, TFORM, ’BICUBIC’, ’FILL’, 0);** % Warps the image according to transformation
## Birdseye View on What We Learned So Far

<table>
<thead>
<tr>
<th>Problem</th>
<th>Detection</th>
<th>Description</th>
<th>Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find Planar Distinctive Objects</td>
<td>Scale Invariant Interest Points</td>
<td>Local feature: SIFT</td>
<td>All features to all features + Affine / Homography</td>
</tr>
<tr>
<td>Panorama Stitching</td>
<td>Scale Invariant Interest Points</td>
<td>Local feature: SIFT</td>
<td>All features to all features + Homography</td>
</tr>
</tbody>
</table>
Exercise: How Dangerous is This Street?

- Can I walk here during the night? Can we tell this from an image?
Exercise: How Dangerous is This Street?

- Can I walk here during the night? Can we tell this from an image?
Exercise: How Dangerous is This Street?

- It's Chicago...

http://www.neighborhoodscout.com/il/chicago/crime/
Exercise: How Dangerous is This Street?

- It’s Chicago... Can I walk here during the day?
Exercise: How Dangerous is This Street?

- Idea: Match image to Google’s StreetView images of Chicago!
Exercise: How Dangerous is This Street?

- Our match to StreetView
Exercise: How Dangerous is This Street?

- Lookup the GPS location...
Exercise: How Dangerous is This Street?

- Lookup the crime map for that GPS location

http://www.neighborhoodscout.com/il/chicago/crime/
Exercise: How Dangerous is This Street?

- Lookup the crime map for that GPS location

http://www.neighborhoodscout.com/il/chicago/crime/
We’re in 2019...

Think not (only) what you can do with one image, but what **lots and lots** of images can do for you
Lesson of the Exercise

- We’re in 2019...

Think not (only) what you can do with one image, but what **lots and lots** of images can do for you

- Would our current matching method work with lots of data?
So far we matched a known object in a new viewpoint

What if we have to match an object to **LOTS** of images? Or **LOTS** of objects to one image?

Please read this and we will discuss:

Josef Sivic, Andrew Zisserman

*Video Google: A Text Retrieval Approach to Object Matching in Videos*

ICCV 2003

Paper link: http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic03.pdf
Next Time:
Camera Models