Panoramas

Calvin and Hobbes

by Watterson

AHHHH...

UH-OH. SOMETHING IS SERiously WRONG HERE.

THE LAWS OF PERSPECTIVE HAVE BEEN REPEALED!

OBJECTS NO LONGER DIMINISH IN SIZE WITH DISTANCE.

LINES DO NOT CONVERGE TOWARD ANY POINT ON THE HORIZON!

ALL SPATIAL RELATIONSHIPS ARE LOST! IT'S IMPOSSIBLE TO JUDGE WHERE ANYTHING IS. OH NO!

CALVIN, QUIT RUNNING AROUND AND CRASHING INTO THINGS, OR I'LL SELL YOU TO THE MONKEY HOUSE!

...AND NOW SHE'S LOST PERSPECTIVE.

Tuesday, October 8, 13
Applied homographies

- http://vimeo.com/75260457
Panoramas and Homographies

Fredo Durand
MIT EECS 6.815/6.865
Motivation
Panorama: virtual wide angle
All the way to 360

- http://people.csail.mit.edu/fredo/Panos/
Similar to HDR

- Medium is limited (range, field of view)
- Take multiple images
- Merge
An old idea

This image by the firm of Maison Bonfils depicts the city of Beirut, Lebanon, sometime in the last third of the 19th century. Maison Bonfils was the extraordinarily prolific venture of the French photographer Félix Bonfils (1831-85), his wife Marie-Lydie Cabanis Bonfils (1837-1918), and their son, Adrien Bonfils (1861-1928). The Bonfils moved to Beirut in 1867 and, over the next five decades, their firm produced one of the world's most important bodies of photographic work about the Middle East. Maison Bonfils was known for landscape photographs, panoramas, biblical scenes, and posed “ethnographic” portraits. The family’s marketing acumen and commercial sense helped make their photographs known around the world. “Panoramic” photographs employ a variety of techniques to create a wide angle of view. This “panoramic view” is composed of four aerial photographs set together to give the viewer a broader image than would have been practical with a single photograph.

http://en.wikipedia.org/wiki/File:%D8%A8%D9%8A%D8%B1%D9%88%D8%AA-%D8%A7%D9%84%D9%82%D8%B1%D9%86_%D9%A1%D9%A9.jpg
An old idea

Old panos, modern viewer

- [http://www.eurofresh.se/history/](http://www.eurofresh.se/history/)

*Figure 2. The principle of an infinite rotation presentation of a circular panorama (circular room or a computer screen and panorama viewer).*
Overview
Today

- The user gives us 4 correspondences
- We reproject one image to match the other one
- Creates a wider angle view
Later

- Automagic correspondences
  - corner detection
  - patch descriptor

- Nice blending
  - smooth transition
  - 2 scale
Overview

• Today: Manual panorama
  – perspective and image reprojection
  – Homogenous coordinates and homographies
  – Inferring homography from correspondences

• Automatic panorama

• Nice blending
Reprojection
Virtual wide angle

• Take N images in different directions
• Deduce the image that would have been taken by a wider angle lens
• …but wait, why should this work at all?
  – What about the 3D geometry of the scene?
  – Why aren’t we using it?
A pencil of rays contains all views

Can generate any synthetic camera view as long as it has the same center of projection!
Entrance pupil

- When camera is rotated around entrance pupil, there is no parallax
  - That is, if two 3D points are superimposed for one orientation, they remain superimposed after rotation

- Finding the entrance pupil is painful
  - and it’s often incorrectly called nodal point

Recap

• When we only rotate the camera (around nodal point) depth does not matter

• It only performs a 2D warp
  – one-to-one mapping of the 2D plane
  – plus of course reveals stuff that was outside the field of view

• Now we just need to figure out this mapping
Aligning images

We have established that pairs of images from the same viewpoint can be aligned through a simple 2D spatial transformation (warp).

What kind of transformation?
Aligning images: translation

Translations are not enough to align the images
Recap

• We are looking for the 2D mapping that corresponds to a 3D rotation of the camera
• We need to understand perspective projection
• There is a simple transformation from \(x,y\) in one image to \(x',y'\) in the other one
• Simple matrix multiplication+ division
• No depth required
Spoiler: simple 2D warp

• For each pixel y,x, in output
  - \((y', x', w') = H^{-1}(y, x, 1)\)
  - \((y'', x'') = (y'/w', x'/w')\)
  - \(\text{out}(y, x) = \text{in}(y'', x'')\)

\(H^{-1}\) matrix multiply division

\((x, y, 1)\) output (in coordinate system of input 1)

\((x', y', w')\) input 2

\((x'/w', y'/w')\)
Goal: cut the (3e) middleman
Projective geometry
Simple Perspective Projection

• Project all points to the $z = 1$ plane, viewpoint at the origin:
  
  $-x' = \frac{x}{z}$
  
  $-y' = \frac{y}{z}$
Simple Perspective Projection

- Project all points to the $z = 1$ plane, viewpoint at the origin:
  - $x' = x/z$
  - $y' = y/z$

- Can we represent this with a matrix?
  - not directly (division)
  - but we can cheat...
  - add a third coordinate to the result
    - interpret as: we always divide by 3rd coordinate
    - see next slide...
3D rotation

- We observe point $x$, $y$, $z$ with camera PP1
  - first image of our pano
3D rotation

- We observe point $x, y, z$ with camera PP1
  - first image of our pano
- Now we rotate the 3D camera to PP2
  - second image of our pano
- What is the new 2D projection?
  - i.e. where does the point from the first image reproject into the second one?
3D rotation

• Rotating the camera is the same as rotating the world in the opposite direction

• To project a (x,y,z) wrt rotated camera:
  – Apply rotation $R^{-1}$ to (x,y, z)
  – Apply projection division
3D rotation: Pano case

- Each image has a different rotation $R_i$
- project $(x, y, z)$ wrt camera rotated by $R_i$:
  - Apply rotation $R_i^{-1}$ to $(x, y, z)$
  - Apply projection division

- Now we understand how to go from the 3D world to individual pano input images
  - We now need to go directly from 2D input to another one
Recap

- canonical projection = division by $z$
- For other direction, just apply 3D rotation first

But... this applies only when we know $z$
  - And for panorama stitching, we don’t
  - What are we going to do? Are we in big trouble? Should we give up? Buy a 3D scanner? Cancel assignment 6?
Unknown z
Points that project on the same x,y
Project to different image?
Rotate
Rotate

\[ z = 1 \]
Divide by $z$
Project to different image

– It doesn’t depend on z
– All points that project to the same x, y reproject onto the same x’, y’
– after perspective division
Back to pano problem

- We want to know how points at $x, y$ in the first image get reprojected onto the rotated second image
  - problem: we don’t know their $z$
Back to pano problem

- We want to know how points at x,y in the first image get reprojected onto the rotated second image
  - problem: we don’t know their z
  - BUT ID DOESN’T DEPEND ON Z!!!!!!
  - LET’S PICK AN ARBITRARY Z=1
Project to different image?
• Apply R to vector \((x, y, 1)\)
Rotate

- Apply R to vector \((x, y, 1)\)
Rotate

- Apply R to vector \((x, y, 1)\)
- divide resulting \(x', y'\) by resulting \(z'\)
Bottomline

• The transformation from one 2D image to a rotated image of the same scene is a $3 \times 3$ matrix $H$ applied to $x,y,1$ followed by a division by $z'$
  – called a homography

\[
H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x'/z' \\ y'/z' \\ 1 \end{pmatrix}
\]

• Now the only problem is that we usually don’t know $H$
Today

• The user gives us 4 correspondences
• Use regression to compute the matrix
Homogeneous coordinates
1D homogeneous coordinates

• Add one dimension to make life simpler
• \((x, w)\) represent point \(x/w\)
Other illustration: 1D homography

- Reproject to different line
1D homography

- Reproject to different line

\[ w=1 \]
1D homography

• Reproject to different line
• Equivalent to rotating 2D points
  ➔ reprojection is linear in homogeneous coordinates
  – and independent of depth
Homogenous coordinates

• Representation of 2D points using 3 coordinates

• \((x, y, w)\) represents \((x/w, y/w)\)
  – Homogenous points are interpreted as Euclidean coordinates by dividing by the last coordinate

• Motivations:
  – perspective transforms
  – Make translations linear
Homogeneous coordinate

- represent 2D points with 3 numbers
- \((x, y, w)\) represents \((x/w, y/w)\)
- Allows us to represent projective transforms
- Nice thing: projecting onto plane \(z=1\) is just the strict interpretation of homogeneous coordinates

- Yes, you can view this as a notation trick
  - But math is all about smart notations
- Homogenous coordinates are central in computer graphics and machine vision
Scale invariant

- All points \((wx, wy, w)\) for any non-zero \(w\) represent the same Euclidean point
Extra profit

• Homogenous coordinates allow us to represent 2D translations as matrices

\[
\begin{pmatrix}
1 & 0 & t_y \\
0 & 1 & t_x \\
0 & 0 & 1
\end{pmatrix}
\]
Questions?

- [http://globalreefrecord.org/home_scientific](http://globalreefrecord.org/home_scientific)
Homographies
2D homogenous version

- 2D points represented as homogenous coordinates \((x, y, 1)\)
- Mapped into the other image by a 3x3 matrix \(H\), into homogenous coordinates \((x’, y’, w’)\)
- get Euclidean coordinates by dividing by \(w’\)
Warping with homography

- For each pixel y,x, in output
  - \((y', x', w') = H^{-1}(y, x, 1)\)
  - \((y'', x'') = (y'/w', x'/w')\)
  - out\((y, x) = in(y'', x'')\)

Important output (in coordinate system of input 1)

Input 2

output
3D vs. homogenous: the same

- 3 coordinates because viewed as plane in 3D
- 3x3 matrix because rotation
- Divide because perspective
- weird extra coordinate
- 3x3 homography matrix
- divide because it’s the rule

projection plane

R(x’,y’, l)
(x”, y”, l)

(x, y, 1)

(x’, y’, w’)
(x’/w’, y’/w’)

H

Tuesday, October 8, 13
Wrapping it up: Homography

- Projective mapping between any two planes
- represented as 3x3 matrix in homogenous coordinates
  - corresponds to the 3D rotation

\[
\begin{bmatrix}
wx' \\
wv' \\
p'
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & * \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

To apply a homography $H$
- Compute $p' = Hp$ (regular matrix multiply)
- Convert $p'$ from homogeneous to image coordinates (divide by $w$)
Wrapping it up: Homography

• rectangles map to arbitrary quadrilateral
• parallel lines aren’t parallel anymore
• but straight lines remain straight

• same as: project, rotate, reproject
Recap

• Reprojection = homography
• 3x3 matrix in homogeneous coordinate
  – (the matrix can be constrained to be a rotation)

\[
\begin{bmatrix}
wx' \\
wy' \\
p',w
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & * \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
l
\end{bmatrix}
\]
Homographies are defined up to a scale

- $H$ and $kH$ represent the same 2D transformation
- because $(w'x', w'y', w')$ and $(kw'x', kw'y', kw')$ represent the same point

\[
\begin{bmatrix}
wx' \\
w'y' \\
p'
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & *
\end{bmatrix} \begin{bmatrix}
x \\
y \\
l
\end{bmatrix}
\]
General pairs of plane

- Homographies can project from any plane to any plane
General pairs of plane

- Homographies can project from any plane to any plane

- Note: here we only warped the image rectangle, but the homography is defined for the whole plane, any x,y

\[
\begin{bmatrix}
0.8346 & -0.0058 & -141.3292 \\
0.0116 & 0.8025 & -78.2148 \\
-0.0002 & -0.0006 & 1
\end{bmatrix}
\]

The function does not return anything, it just modifies output.

We will treat pixels coordinates outside the source image differently from previous problem sets. For compositing, we want to leave the corresponding output pixels untouched. For this, you need to test if the warped coordinates are outside the source image before updating the pixel value. This scheme might lead to stair artifacts at the boundary but we won’t worry about it. In a later assignment, we will perform nice feathering at the boundaries between images.

Test your function on the provided `green.png` and `poster.png` images using the homography: $H = \begin{bmatrix} 0.8346 & -0.0058 & -141.3292 \\ 0.0116 & 0.8025 & -78.2148 \\ -0.0002 & -0.0006 & 1 \end{bmatrix}$

Solve for a homography from four pairs of points

In this section, a user will provide correspondences between two images, by clicking using a patent-pending javascript interface, and your job is to infer the homography matrix that maps the points from the first image into those in the second one.

The correspondences will be provided as a list of pairs of points, where each point is encoded as an array of size 2 that contains the y and x coordinates of a point. The order of coordinates is y,x, unlike problem set 2, which should make your life easy. Indexing will be according to `listOfPairs[pair index][image][coordinate]`. This means that `listOfPairs[2][0][0]` contains the y coordinate in the first image for the third pair. Its y coordinate in the second image is `listOfPairs[2][1][0]`.

Like for problem set 2, you need to click on points in the same order for the left and right images.

Write a function `computehomography(listOfPairs)` that takes a list of pairs of points and return a 3x3 homography matrix that maps the first point of each pair to the second one. That is, the homography maps from the first image to the second one.
Image warping with homographies

homography so that image is parallel to floor

homography so that image is parallel to right wall

black area where no pixel maps to
Questions?

Solving for homographies
Goal

- Given correspondences
- Find homography matrix $H$ that maps the $p_i$ to $p_i'$
Warning

• In what follows I use the order y, x
  – At least I will try
• This will make life easier for pset 6
Homography equation

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

- We are given pairs of corresponding points
  - \(x, y, x', y'\) are known
- Unknowns: matrix coefficients and \(w'\)
Homography equation

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

• We are given pairs of corresponding points
  – x, y, x’, y’ are known
• Unknowns: matrix coefficients and w’
  – but w’ is easy to get:
    \[ w' = gy + hx + i \]
\[
\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}
\]

\[w' = gy + hx + i\]

- For a pair of points \((x, y)\rightarrow(x', y')\) we have

\[ay + bx + c = y'(gy + hx + i)\]

\[dy + ex + f = x'(gy + hx + i)\]
For a pair of points \((x, y)\rightarrow(x', y')\) we have

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

\[w' = gy + hx + i\]

- **For a pair of points** \((x, y)\rightarrow(x', y')\) we have
  \[
  ay + bx + c = y'(gy + hx + i)
  \]
  \[
  dy + ex + f = x'(gy + hx + i)
  \]

- **Unknowns:** \(a, b, c, d, e, f, g, h, i\)
  - **Linear!**
How many pairs?

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

- Each correspondence pair gives us two equations

\[
ay + bx + c = y'(gy + hx + i)
\]
\[
dy + ex + f = x'(gy + hx + i)
\]

- How many unknowns?
  - 9
  - but H is defined up to scale. Four pairs are
Forming the linear system

- We have 4x2 linear equations in our 8 unknowns
- Represent as a matrix system $Ax = B$:

\[
\begin{pmatrix}
A & \begin{pmatrix}
a \\
b \\
c \\
d \\
e \\
f \\
g \\
h \\
i
\end{pmatrix}
\end{pmatrix}
= B
\]

- Now we need to fill matrix $A$ and vector $B$
Forming the matrix

\[ ay + bx + c = y' (gy + hx + i) \]
\[ dy + ex + f = x' (gy + hx + i) \]
Forming the matrix

\[ ay + bx + c = y'(gy + hx + i) \]
\[ dy + ex + f = x'(gy + hx + i) \]

\[
\begin{pmatrix}
  a & b & c & d & e & f & g & h & i \\
  y & x & l & 0 & 0 & 0 & -yy' & -xy' & -y' \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
  g \\
  h \\
  i \\
\end{pmatrix}
\]

= 0

I’ll let you do the x case for pset 6
Recap

• We have four pairs of points

• Looking for homography H

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

• Formed a big 8x9 linear system \( Ax = 0 \)
  – where \( x \) is the 9 homography coefficients
Solve for scale invariance

• We know that there exists a full family of solutions $kH$, for any non-zero $k$

• i.e., if $(a, b, c, d, e, f, g, h, i)$ is a solution, so is $(ka, kb, kc, kd, ke, kf, kg, kh, ki)$

• Adding more correspondences won’t help
Dirty solution

• Hope that $i$ is not 0
• Set it arbitrarily to 1
  – either create an 8x8 matrix
  – or add a last row that says $i$ should be 1

• In practice $i$ is rarely zero
  – But it’s still dirty

• You can use this solution for pset 6
Cleaner solution

• Use SVD
• The singular vector with singular value 0 is a solution

• See your favorite linear algebra textbook
Dirty-clean: Three versions

• Dirty with $i=1$
  – $9 \times 9$:
    – RHS is zero almost everywhere, except last row
    – last row just says $i=1$
  – $8 \times 8$:
    – substitute $i=1$ from beginning
    – RHS is usually non-zero

• Clean with SVD: $8 \times 9$
  – $Ax=0$
Questions?

• Julian Beever
• e.g. http://users.skynet.be/J.Beever/
  http://www.crystalinks.com/julian_beever.html
Demo

- [http://openphotovr.org/edit.html?id=33RwaQHm](http://openphotovr.org/edit.html?id=33RwaQHm)
- If we change the p’, we get a different homography
Homography

• Different sets of $p$, $p'$ correspondences can yield the same homography
Homography

- Different sets of \( p, p' \) correspondences can yield the same homography
In pset 5

- First use to warp from a plane to a 3D plane

- Then use to warp pano inputs
Metaphor

- Example of fitting lines, $p' = ap + b$, different pairs of correspondences ($p_1 \rightarrow p'_1$, $p_2 \rightarrow p'_2$) define the same line.
Beware

• Homographies are defined for the whole plane – just apply the formula!
• But we often only want to apply it inside the rectangle of the input

• And as usual, recall forward vs. inverse warp
Recap
Manual linear panoramas

• Create a virtual wide angle view from 2 images
• Choose one image as reference
• User gives 4 correspondences
• Deduce homography matrix
• Reproject (warp) second image into first one
Under the hood

• Homogenous coordinates
  – encode 2D points with 3 coordinates (x, y, w)
  – represents Euclidean points (x/w, y/w)
  – make it easy to express perspective and to go between 3D and 2D

• Homography
  – 3x3 matrix on homogenous coordinates
  – represent any perspective mapping of a plane
  – to apply a homography to a 2D point x, y: compute $H(x, 1, y)^\top$ and divide by third
Multiple images

1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend
Homography warp

• For each output pixel
• compute input location with homography matrix
  — \( p' = Hp \), followed by division
• copy pixel color (with appropriate antialiasing)
Later

- Automagic correspondences
  - corner detection
  - patch descriptor

- Nice blending
  - smooth transition
  - 2 scale

- Other projections
  - spherical, cylindrical, miniplanets

http://designedbynatalie.com/tag/panoramas/
Magic: automatic panos

Questions?
changing camera center

• Does it still work?
## Only for Planar mosaic

<table>
<thead>
<tr>
<th>Image Compositing for Tele-Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>2. Previous work</td>
</tr>
<tr>
<td>3. Basic imaging equations</td>
</tr>
<tr>
<td>4. Planar image compositing</td>
</tr>
<tr>
<td>5. Panoramic</td>
</tr>
<tr>
<td>6. Piecewise-planar scenes</td>
</tr>
<tr>
<td>7. Scenes with arbitrary depth</td>
</tr>
<tr>
<td>8. 3-D model recovery</td>
</tr>
<tr>
<td>9. Applications</td>
</tr>
<tr>
<td>10. Discussion &amp; conclusions</td>
</tr>
</tbody>
</table>

... composed scenes...?
Questions?
Applied homographies

- [https://www.youtube.com/watch?feature=player_embedded&v=tBNHPk-Lnkk#](https://www.youtube.com/watch?feature=player_embedded&v=tBNHPk-Lnkk#)
Perspective correction
Digression: perspective correction

• Perspective makes parallel lines converge
• Sometimes objectionable
• In particular, architecture photography
  – photo looks more formal if verticals stay vertical

• But if lines are parallel to the image plane, parallel lines don’t converge
Standing at street level and shooting straight at a building produces too much street and too little building. Sometimes it is possible to move back far enough to show the entire building while keeping the camera level, but this adds even more foreground and usually something gets in the way.

From Photography, London et al.
Standing at street level and shooting straight at a building produces too much street and too little building. Sometimes it is possible to move back far enough to show the entire building while keeping the camera level, but this adds even more foreground and usually something gets in the way.

Tilting the whole camera up shows the entire building but distorts its shape. Since the top is farther from the camera than the bottom, it appears smaller; the vertical lines of the building seem to be coming closer together, or converging, near the top. This is named the keystone effect, after the wedge-shaped stone at the top of an arch. This convergence gives the illusion that the building is falling backward—an effect particularly noticeable when only one side of the building is visible.

From Photography, London et al.
Standing at street level and shooting straight at a building produces too much street and too little building. Sometimes it is possible to move back far enough to show the entire building while keeping the camera level, but this adds even more foreground and usually something gets in the way.

Tilting the whole camera up shows the entire building but distorts its shape. Since the top is farther from the camera than the bottom, it appears smaller; the vertical lines of the building seem to be coming closer together, or converging, near the top. This is named the keystone effect, after the wedge-shaped stone at the top of an arch. This convergence gives the illusion that the building is falling backward—an effect particularly noticeable when only one side of the building is visible.

To straighten up the converging vertical lines, keep the camera back parallel to the face of the building. To keep the face of the building in focus, make sure the lens is parallel to the camera back. One way to do this is to level the camera and then use the rising front or falling back movements or both.

Another solution is to point the camera upward toward the top of the building, then use the tilting movements—first to tilt the back to a vertical position (which squares the shape of the building), then to tilt the lens so it is parallel to the camera back (which brings the face of the building into focus). The lens and film will end up in the same positions with both methods.
• Lecture stopped here.
• Equivalent to using a wider-angle view and cropping

From Photography, London et al.
Perspective correction

- The lens creates an image bigger than the sensor
- Tilting and shifting aligns the sensor with the part we want
• The two images are related by a homography

Tilting the whole camera up shows the entire building but distorts its shape. Since the top is farther from the camera than the bottom, it appears smaller; the vertical lines of the building seem to be coming closer together, or converging, near the top. This is named the keystone effect, after the wedge-shaped stone at the top of an arch. This convergence gives the illusion that the building is falling backward—an effect particularly noticeable when only one side of the building is visible.

To straighten up the converging vertical lines, keep the camera back parallel to the face of the building. To keep the face of the building in focus, make sure the lens is parallel to the camera back. One way to do this is to level the camera and then use the rising front or falling back movements or both.

Another solution is to point the camera upward toward the top of the building, then use the tilting movements—first to tilt the back to a vertical position (which squares the shape of the building), then to tilt the lens so it is parallel to the camera back (which brings the face of the building into focus). The lens and film will end up in the same positions with both methods.

From Photography, London et al.
Tilt-shift lens

- 35mm SLR version
Photoshop version (perspective crop)

+ you control reflection and perspective independently
Question?

- The Ambassadors, Hans Holbein the Younger, 1533
Question?

- The Ambassadors, Hans Holbein the Younger, 1533
Cool applications of homographies

• With Mok Oh.
Limitations of 2D Clone Brushing

• Distortions due to foreshortening and surface orientation
Clone brush (Photoshop)

• Click on a reference pixel (blue)
• Then start painting somewhere else
• Copy pixel color with a translation
Perspective clone brush

Oh, Durand, Dorsey, unpublished

• Correct for perspective
• And other tricks
Figure 15: The cars and the street furniture have been removed. This example took less than 10 minutes.
Questions?
Other application: View morphing

- We want to morph between two views of the same object
- Standard morphing won’t give a realistic result
  - Because it uses linear interpolation of image locations
  - With perspective & 3D, points don’t move linearly

Figure 2: A Shape-Distorting Morph. Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques.
View morphing

- Seitz & Dyer
- Interpolation consistent with 3D view interpolation

Figure 1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.
View morphing

- Given view 1 and view 2 (and no depth!)
- Compute intermediate views consistent with 3D

subject

viewpoint 1

viewpoint 2
View morphing

- Reproject to a common view plane parallel to the 2 viewpoints: homographies
View morphing

• Similar triangles: everything becomes linear
• i.e. interpolate location of features linearly
View morphing

- Similar triangles: everything becomes linear
- Independent of depth

subject

Q1 Q2

common view plane

viewpoint 1 viewpoint 2
View morphing recap

- When views are reprojected onto a common plane parallel to the line between the viewpoints: Interpolating the viewpoint results in linear interpolation of feature locations

- i.e.
  - for viewpoint \((1-t)V_1 + tV_2\)
  - a 3D point that was at \(P_1\) from \(V_1\) and \(P_2\) from \(V_2\) is now at \((1-t)P_1 + tP_2\)

- That is, a simplistic warp works
View morphing

- Prewarp with a homography to "pre-align" images
- So that the two views are parallel
  - Because linear interpolation works when views are parallel

Figure 4: View Morphing in Three Steps. (1) Original images $\mathcal{I}_0$ and $\mathcal{I}_1$ are prewarped to form parallel views $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. (2) $\hat{\mathcal{I}}_s$ is produced by morphing (interpolating) the prewarped images. (3) $\hat{\mathcal{I}}_s$ is postwarped to form $\mathcal{I}_s$. 
Figure 6: View Morphing Procedure: A set of features (yellow lines) is selected in original images $I_0$ and $I_1$. Using these features, the images are automatically prewarped to produce $\hat{I}_0$ and $\hat{I}_1$. The prewarped images are morphed to create a sequence of in-between images, the middle of which, $\hat{I}_{0.5}$, is shown at top-center. $\hat{I}_{0.5}$ is interactively postwarped by selecting a quadrilateral region (marked red) and specifying its desired configuration, $Q_{0.5}$, in $I_{0.5}$. The postwarps for other in-between images are determined by interpolating the quadrilaterals (bottom).
Figure 10: Image Morphing Versus View Morphing. Top: image morph between two views of a helicopter toy causes the in-between images to contract and bend. Bottom: view morph between the same two views results in a physically consistent morph. In this example the image morph also results in an extraneous hole between the blade and the stick. Holes can appear in view morphs as well.
Figure 7: Facial View Morphs. Top: morph between two views of the same person. Bottom: morph between views of two different people. In each case, view morphing captures the change in facial pose between original images $I_0$ and $I_1$, conveying a natural 3D rotation.
Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and its reflection (right).
Extensions

- Video
- Additional objects
- Mok’s panomorph
• Dealing with parallax
Software

- http://photocreations.ca/collage/circle.jpg
- http://webuser.fh-furtwangen.de/%7Edersch/
- http://www.ptgui.com/
- http://hugin.sourceforge.net/
- http://epaperpress.com/ptlens/

http://www.fdrtools.com/front_e.php
• http://www.cs.washington.edu/education/courses/csep576/05wi/readings/szeliskiShum97.pdf
• http://portal.acm.org/citation.cfm?id=218395&dl=ACM&coll=portal
• http://research.microsoft.com/~brown/papers/cvpr05.pdf
• http://citeseer.ist.psu.edu/mann94virtual.html
• http://grail.cs.washington.edu/projects/panovidtex/
• http://research.microsoft.com/vision/visionbasedmodeling/publications/Baudisch-OZCHI05.pdf
• http://www.vision.caltech.edu/lihi/Demos/SquarePanorama.html
• http://graphics.stanford.edu/papers/multi-cross-slits/