Homography

COMP4102A

Dr. Gerhard Roth

Winter 2014

Version 1

Linear Mappings

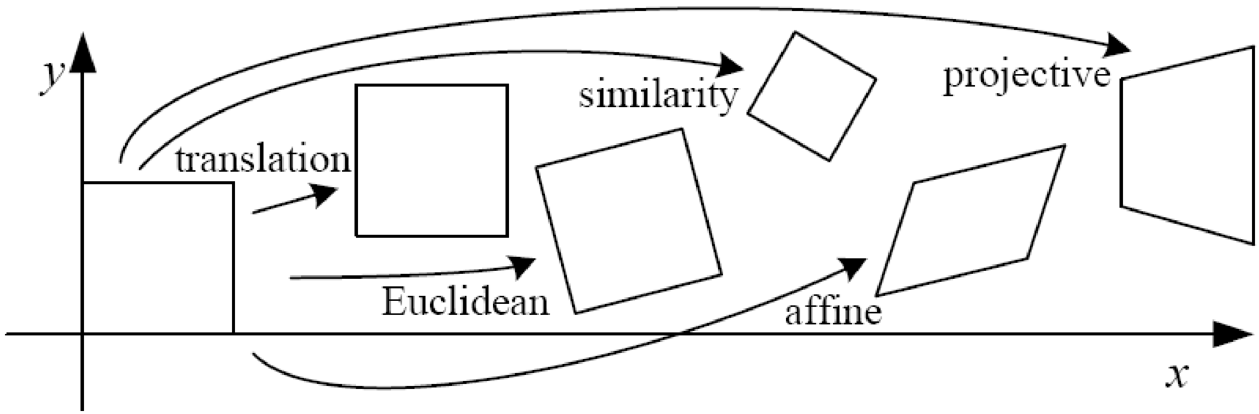
- linear mappings from one particular space to the other, also called transformations
- Different types of transformations from a 2d image to another 2d images
  - Translation, rigid, similarity, affine, projective
  - Next one in the list includes all the previous members
  - i.e. Perspective transformation is a superset of all previous
- A 2d perspective transformation is commonly called an image warp or a homography
- Warp takes a 2d image and produces another 2d image
  - Maps from a pixel in source image to a pixel destination

Homography = Linear warp

- Consider a point x = (u,v,1) in one image and x'=(u',v',1) in another image
- A homography is a 3 by 3 matrix M
- $$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$
- The homography relates the pixel co-ordinates in the two images if x' = M x
    - Works on pixel co-ordinates (pixels) not camera co-ordinates (mm)
- When applied to every pixel the new image is a warped version of the original image

Two images related by homography

2D image transformations

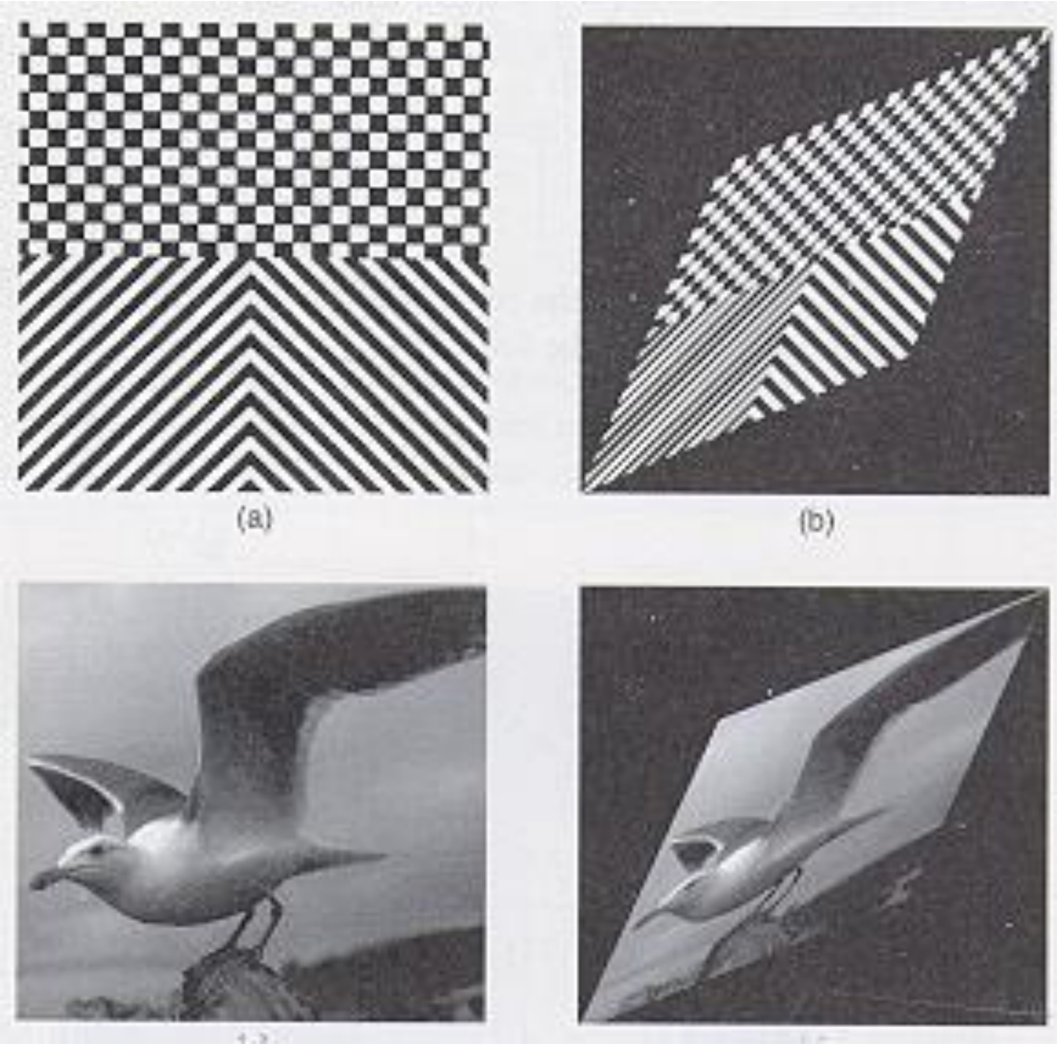


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} & \big\vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} & \big\vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
similarity	$\begin{bmatrix} s\boldsymbol{R} & \big\vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$			
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$			

These transformations are a nested set of groups

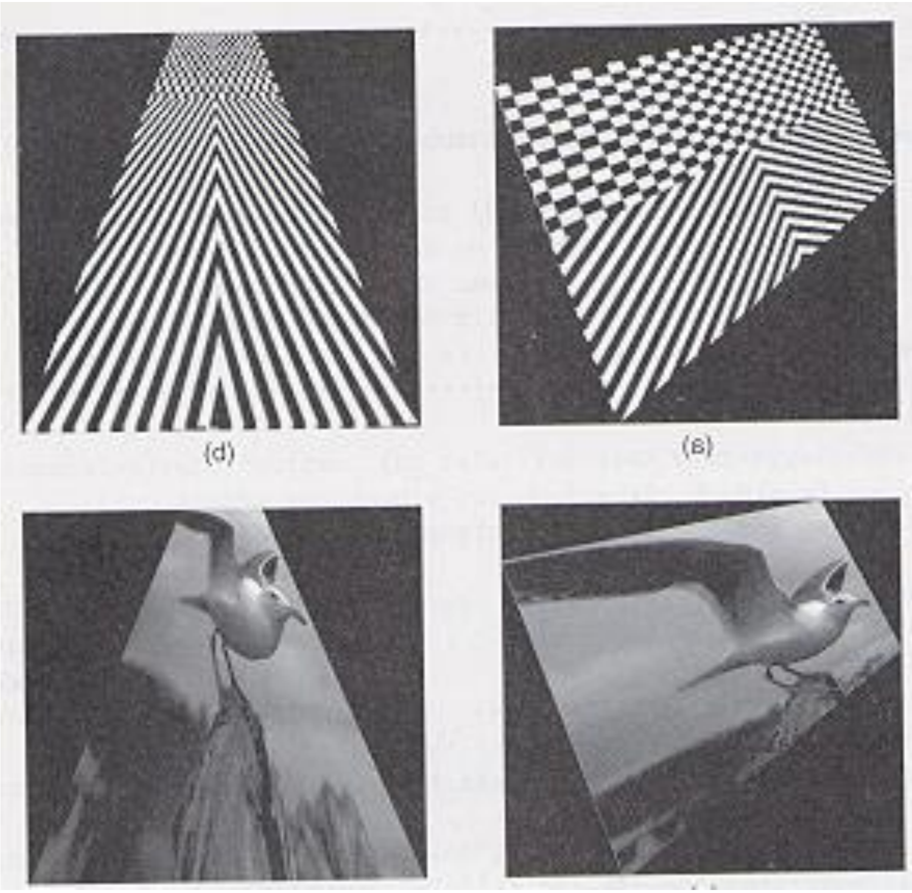
- Closed under composition and inverse is a member

Affine example



Perspective transformation

a straight line is a straight line



Homography conditions

- Two images are related by a homography if and only if (2 conditions)
- Both images are viewing the same plane from a different angle (was on your assignment)
- Both images are taken from the same camera but from a different angle
  - Camera is rotated about its center of projection without any translation
- Note that the homography relationship is independent of the scene structure
  - It does not depend on what the cameras are looking at
  - Relationship holds regardless of what is seen in the images

Homography does not work in general!

Homography when viewing a plane

- Normal projection matrix

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{32} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- All world points on a plane, choose Z = 0 to be on the plane

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & 0 & m_{14} \\ m_{21} & m_{22} & 0 & m_{24} \\ m_{31} & m_{32} & 0 & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

- Final matrix relates points on world plane to image plane

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{14} \\ m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Homography when rotating camera

- No translation, here X is a 3d point, and x,x' projection
- Home position – projection equation

$$\mathrm{x} = \mathrm{K}\left[\mathrm{I} \mid 0\right]\begin{bmatrix} \mathrm{X} \\ 1 \end{bmatrix} = \mathrm{KX}$$

- Rotation by a matrix R – projection equation

$$\mathrm{x}' = \mathrm{K}\left[\mathrm{R} \mid 0\right]\begin{bmatrix} \mathrm{X} \\ 1 \end{bmatrix} = \mathrm{KRX}$$

- So $\mathrm{x}' = \mathrm{KRK}^{-1}\mathrm{x}$ where K is calibration matrix
- Where KRK^{-1} is a 3by3 matrix M called a homography

•Compute homography

- If we know rotation K, R, then homography H can be computed directly $x' = KRK^{-1}x$
  - Applying this homography to one image gives image that we would get if the camera was rotated by R
- If we know, K, R and T and are looking at a plane we can also compute the homography
  - Just look at your last assignment, where you have the full projection matrix (p. 134 of book)
  - Now drop the 3d column, then we have a homography matrix
  - If we know fx, fy, ox, oy, R and T we can define this matrix
- So if we have this info we can always compute the homography
- But what if we only have two images, and a set of correspondences?

Computing M: The four point Algorithm

Input: n point correspondences (n >= 4)

- Construct homogeneous system Ax= 0 from $x' = xM$
  - $x = (m_{11}, m_{12}, , m_{13}, m_{21}, m_{22}, m_{23}\ m_{31}, m_{32}, m_{33})$: entries in m
  - Each correspondence gives two equations
  - A is a 2xnx9 matrix
- Obtain estimate M by Eigenvector with smallest eigenvalue
  - Do not have any singularity constraint as we did with the fundamental matrix
  - Finding the homography that solves Ax = 0 subject to the constraint that
  - x is a unit vector ||x|| =1

Output: the homography matrix, M that is the best least squares solution to this problem

•Similar to how you compute a projection matrix, but the correspondences are 2d pixels to 2d pixels, and not 3d points to 2d pixels!

Recover Homography matrix from correspondences

$$\begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix}$$

$$u_2 = \frac{H_{11} u_1 + H_{12} v_1 + H_{13}}{H_{31} u_1 + H_{32} v_1 + H_{33}}$$

$$v_2 = \frac{H_{21} u_1 + H_{22} v_1 + H_{23}}{H_{31} u_1 + H_{32} v_1 + H_{33}}$$

Here u = x, and v = y

Multiply up denominators

$$H_{31} u_1 u_2 + H_{32} v_1 u_2 + H_{33} u_2 = H_{11} u_1 + H_{12} v_1 + H_{13}$$
$$H_{31} u_1 v_2 + H_{32} v_1 v_2 + H_{33} v_2 = H_{21} u_1 + H_{22} v_1 + H_{23}$$

Solving for AX=0 -gives us 2 equations/correspondence –need 4 correspondences

$$H_{31} u_1 u_2 + H_{32} v_1 u_2 + H_{33} u_2 - H_{11} u_1 - H_{12} v_1 - H_{13} = 0$$
$$H_{31} u_1 v_2 + H_{32} v_1 v_2 + H_{33} v_2 - H_{21} u_1 - H_{22} v_1 - H_{23} = 0$$

Each correspondence provides two rows of A matrix

$$\begin{matrix} [-u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1 u_2 & v_1 u_2 & u_2\,] \\ [\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1 v_2 & v_1 v_2 & v_2\,] \end{matrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ \cdots \end{bmatrix}$$

Recover Homography matrix from correspondences

Each correspondence provides two rows of A matrix

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1 u_2 & v_1 u_2 & u_2 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1 v_2 & v_1 v_2 & v_2 \end{bmatrix}$$

8 equations, 9 unknowns (from 4 correspondences)

$$\begin{bmatrix} -u_{11} & -v_{11} & -1 & 0 & 0 & 0 & u_{11}u_{21} & v_{11}u_{21} & u_{21} \\ 0 & 0 & 0 & -u_{11} & -v_{11} & -1 & u_{11}v_{21} & v_{11}v_{21} & v_{21} \\ -u_{12} & -v_{12} & -1 & 0 & 0 & 0 & u_{12}u_{22} & v_{12}u_{22} & u_{22} \\ 0 & 0 & 0 & -u_{12} & -v_{12} & -1 & u_{12}v_{22} & v_{12}v_{22} & v_{22} \\ -u_{13} & -v_{13} & -1 & 0 & 0 & 0 & u_{13}u_{23} & v_{13}u_{23} & u_{23} \\ 0 & 0 & 0 & -u_{13} & -v_{13} & -1 & u_{13}v_{23} & v_{13}v_{23} & v_{23} \\ -u_{14} & -v_{14} & -1 & 0 & 0 & 0 & u_{14}u_{24} & v_{14}u_{24} & u_{24} \\ 0 & 0 & 0 & -u_{14} & -v_{14} & -1 & u_{14}v_{24} & v_{14}v_{24} & v_{24} \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} = 0$$

AX=b **X**=column vector of homography matrix elements
Get last vector using SVD

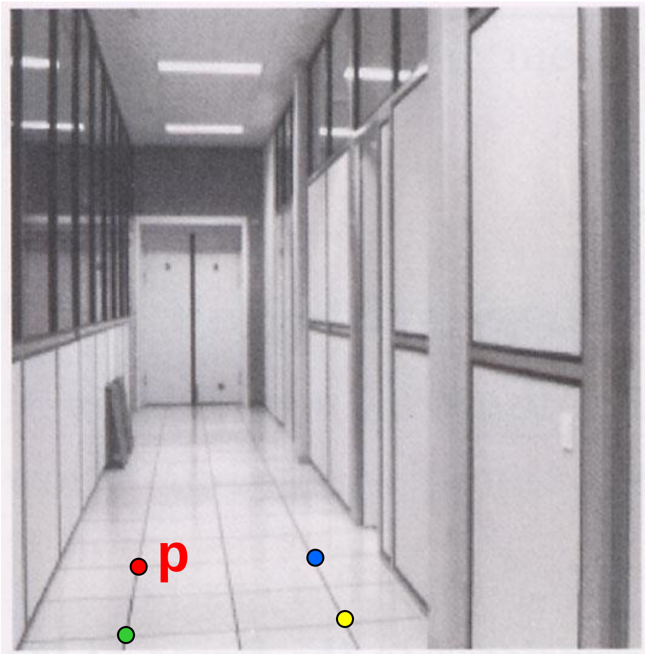
Homogeneous System

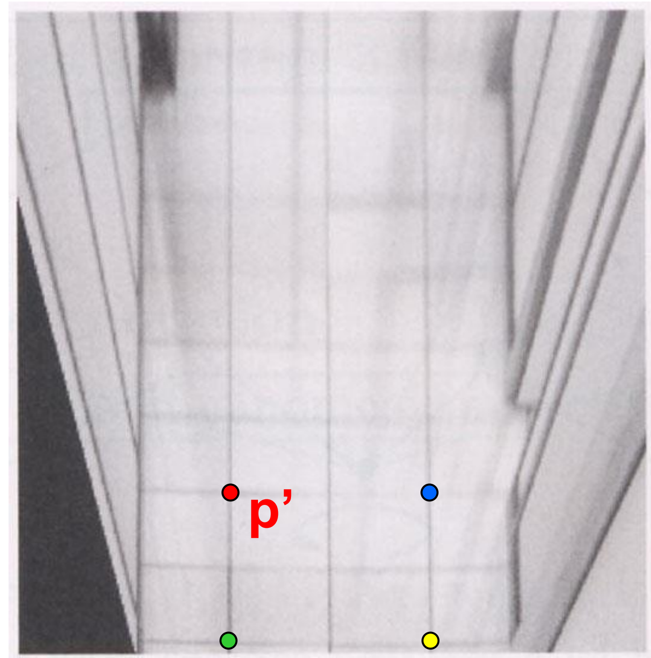
- M linear equations of form A**x** = 0
- If we have a given solution x1, s.t. Ax1 = 0 then c * x1 is also a solution A(c* x1) = 0
- Need to add a constraint on **x**,
  - Basically make **x** a unit vector $x^T x = 1$
- Can prove that the solution is the eigenvector corresponding to the single zero eigenvalue of that matrix $A^T A$
  - This can be computed using eigenvector of SVD routine
  - Then finding the zero eigenvalue (actually smallest)
  - Returning the associated eigenvector

Uses of a homography

- Consider two images (two different views of a plane, or two different rotated views)
- We can synthesize second image from first!
- Take the first image
  - With appropriate homography we can create equivalent of the second image without needing to take that image!
  - In other words, just compute the proper homography and then apply it to the first image to get a new image
  - This new image is the image we would get if we actually physically captured the second image
  - Can not do this in general (only when we are not looking at a plane or when we translate)
- What can we do with this fact?

Image rectification (square views)



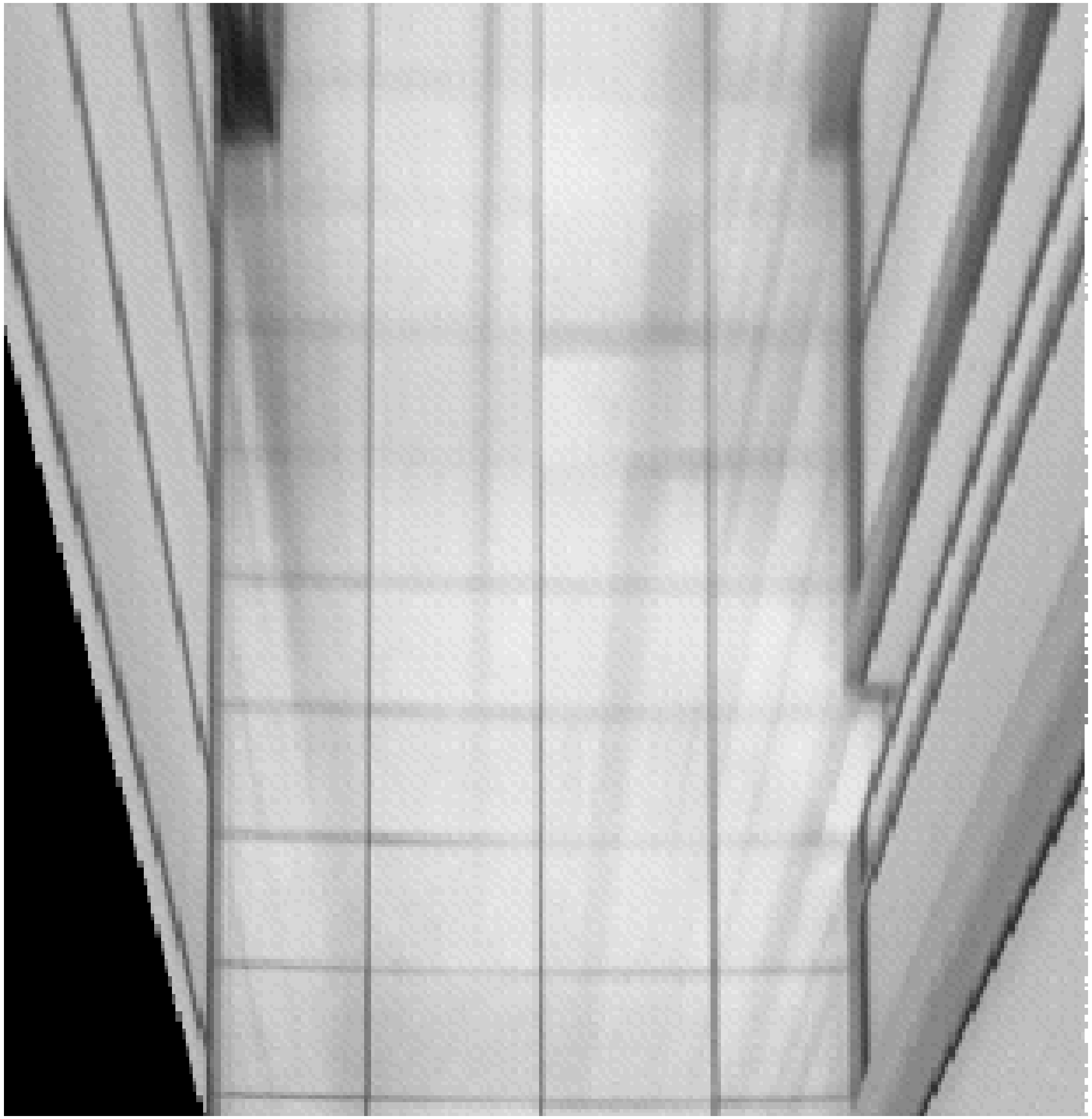


To unwarp (rectify) an image

- Find the homography **H** given a set of **p** and **p’** pairs
- How many correspondences are needed?
- Tricky to write H analytically, but we can <u>solve</u> for it!
  - Find such H that “best” transforms points p into p’
  - Use least-squares!

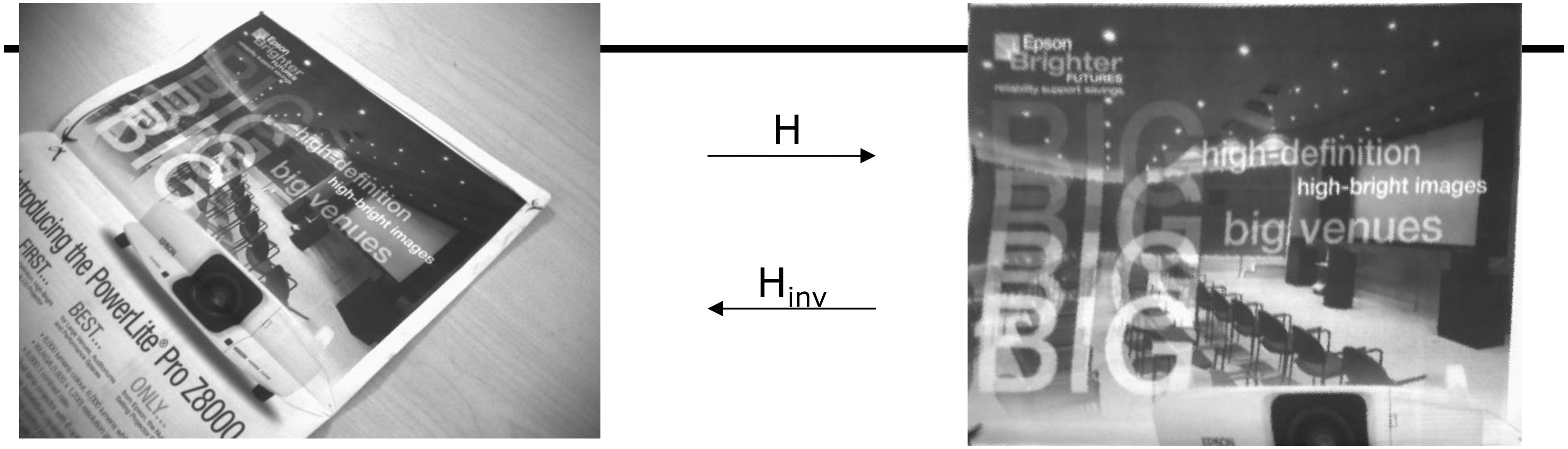
Original camera view

Downward rotation via homography



Sideways rotation via homography

Image Rectification using a homography matrix (from Lec 8)



Correspondences

```
u11=268;  v11=10;   u21=0;    v21=0;
u12=558;  v12=220;  u22=499;  v22=0;
u13=46;   v13=152;  u23=0;    v23=399;
u14=334;  v14=442;  u24=499;  v24=399;
```


```
u11=268;  v11=10;   u21=0;    v21=0;
u12=558;  v12=220;  u22=499;  v22=0;
u13=46;   v13=152;  u23=0;    v23=399;
u14=334;  v14=442;  u24=499;  v24=399;
```


```
a1 =[-u11, -v11, -1, 0, 0, 0,  u11*u21, v11*u21, u21];
a2 =[0, 0, 0,  -u11, -v11, -1, u11*v21, v11*v21, v21];
a3 =[-u12, -v12, -1, 0, 0, 0,  u12*u22, v12*u22, u22];
a4 =[0, 0, 0,  -u12, -v12, -1, u12*v22, v12*v22, v22];
a5 =[-u13, -v13, -1, 0, 0, 0,  u13*u23, v13*u23, u23];
a6 =[0, 0, 0,  -u13, -v13, -1, u13*v23, v13*v23, v23];
a7 =[-u14, -v14, -1, 0, 0, 0,  u14*u24, v14*u24, u24];
a8 =[0, 0, 0,  -u14, -v14, -1, u14*v24, v14*v24, v24];
```

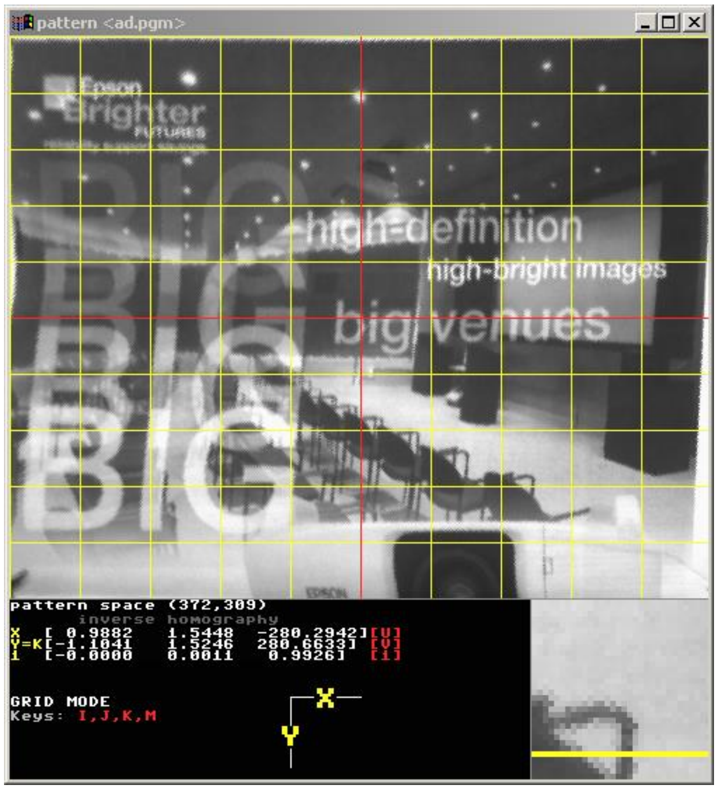
```
[u,d,v]=svd(a)

xtemp=v(8*9+1:9*9)'
x=xtemp/xtemp(9)
H=[x(1),x(2),x(3);x(4),x(5),x(6);x(7),x(8),x(9)]
```

```
H =

    0.9956    1.5566 -282.3961
   -1.1124    1.5362  282.7675
   -0.0000    0.0011    1.0000


Hinv =

    0.3762   -0.5720  268.0000
    0.3401    0.3042   10.0000
   -0.0004   -0.0003    1.0000
```

See matlab_lec9_solve_for_homog_matrix.txt

view_homog_matrix.exe

$$H = \begin{bmatrix} 0.3762 & -0.5720 & 268.0000 \\ 0.3401 & 0.3042 & 10.0000 \\ -0.0004 & -0.0003 & 1.0000 \end{bmatrix}$$



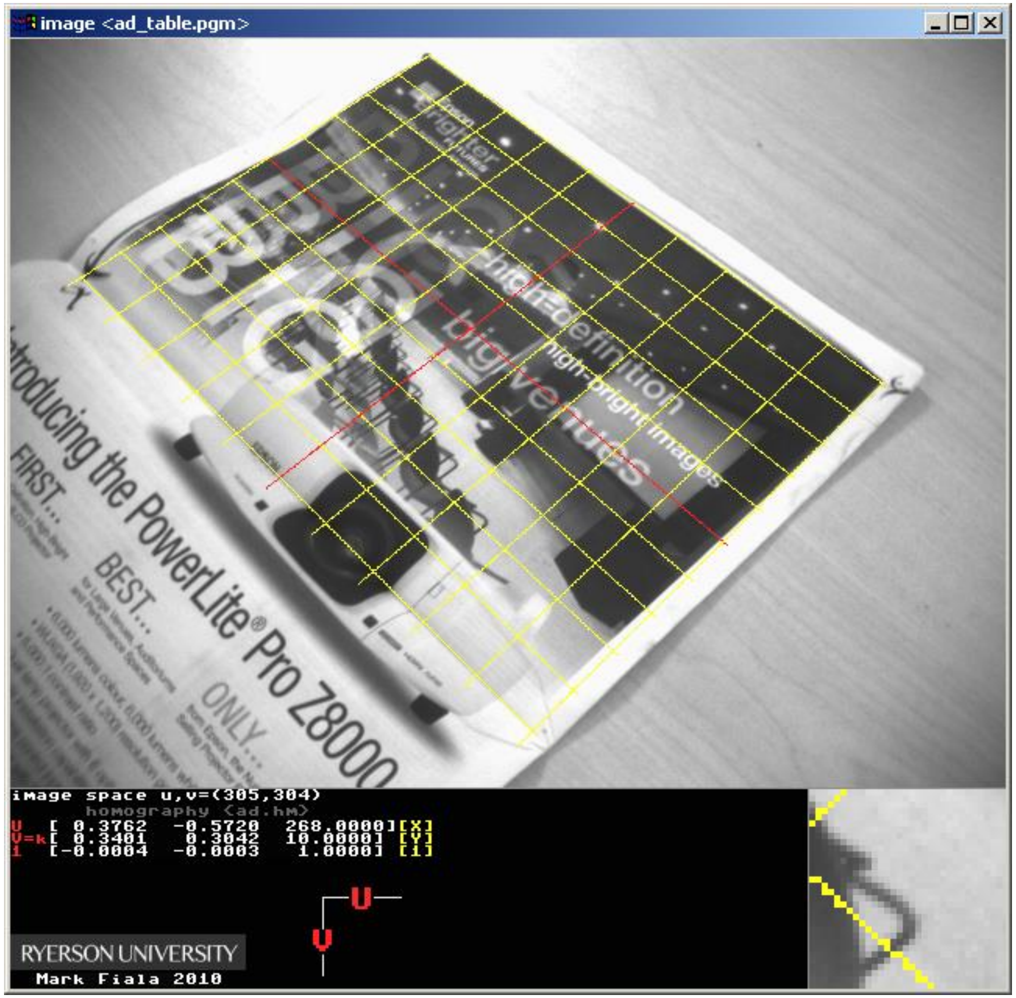
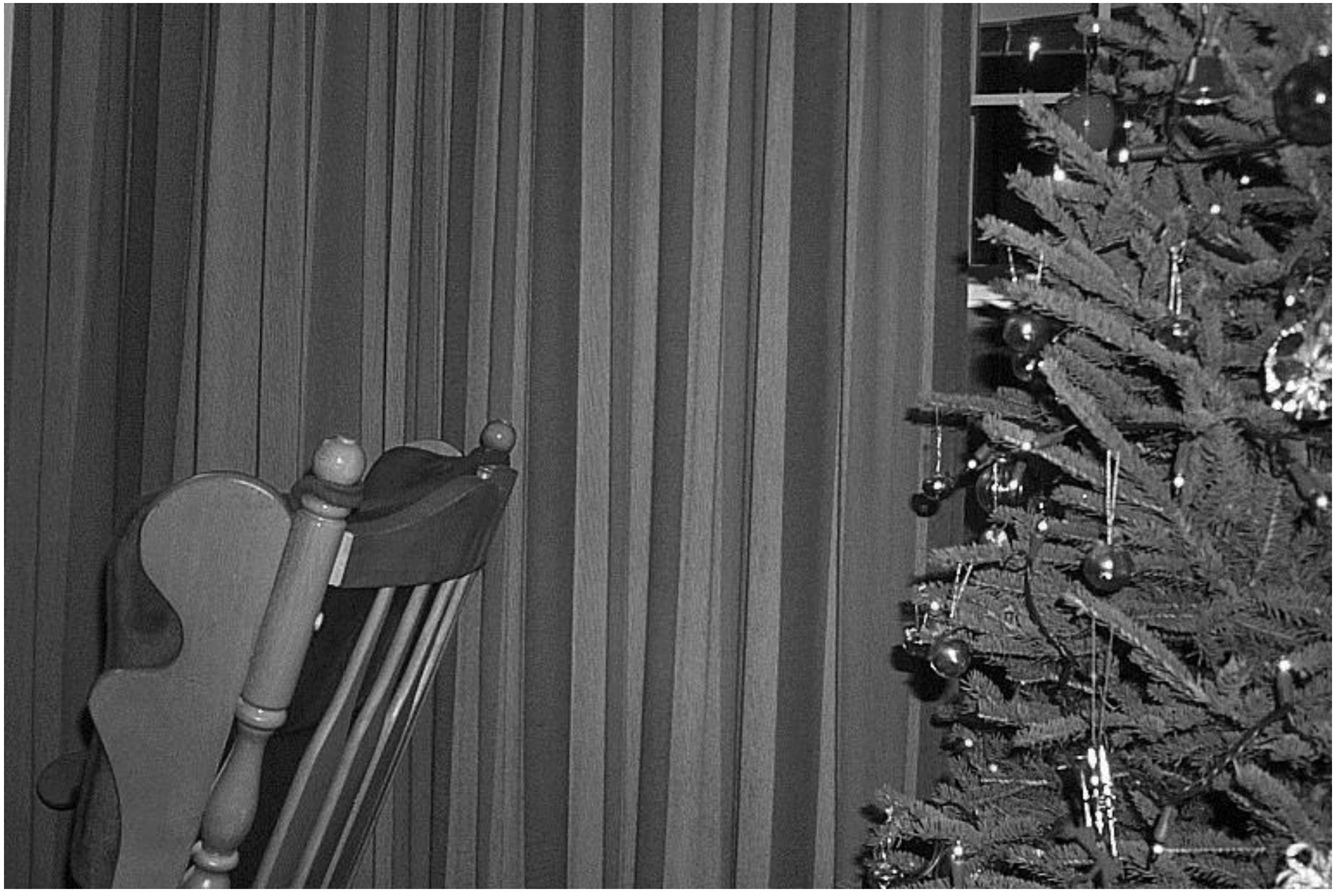


Image Rectification

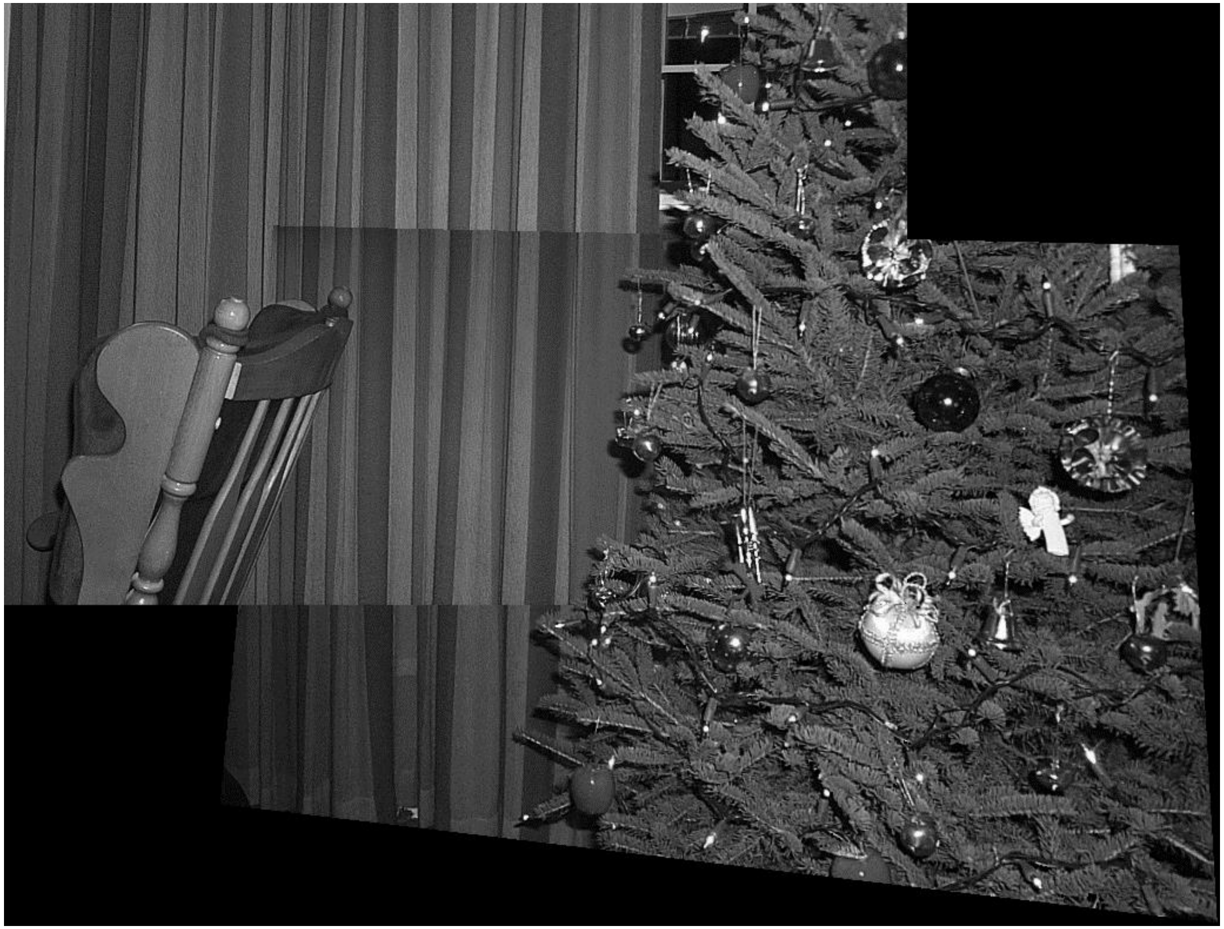
- What are the uses of image rectification
- For a rectified image it is easy to compute object dimensions (given a known dimension
  - Can not do this if the image is not rectified (even if you know the dimensions of an object in the image
- A rectified image is straight on to the camera (along the z axis)
  - This geometry makes matching between two such images easier (called simple stereo matching)
  - Rectification is often done with two images in general position to place them in this standard position
  - Then stereo matching is much easier (will discuss this in the stereo section)

Automatic Mosaicing – Input

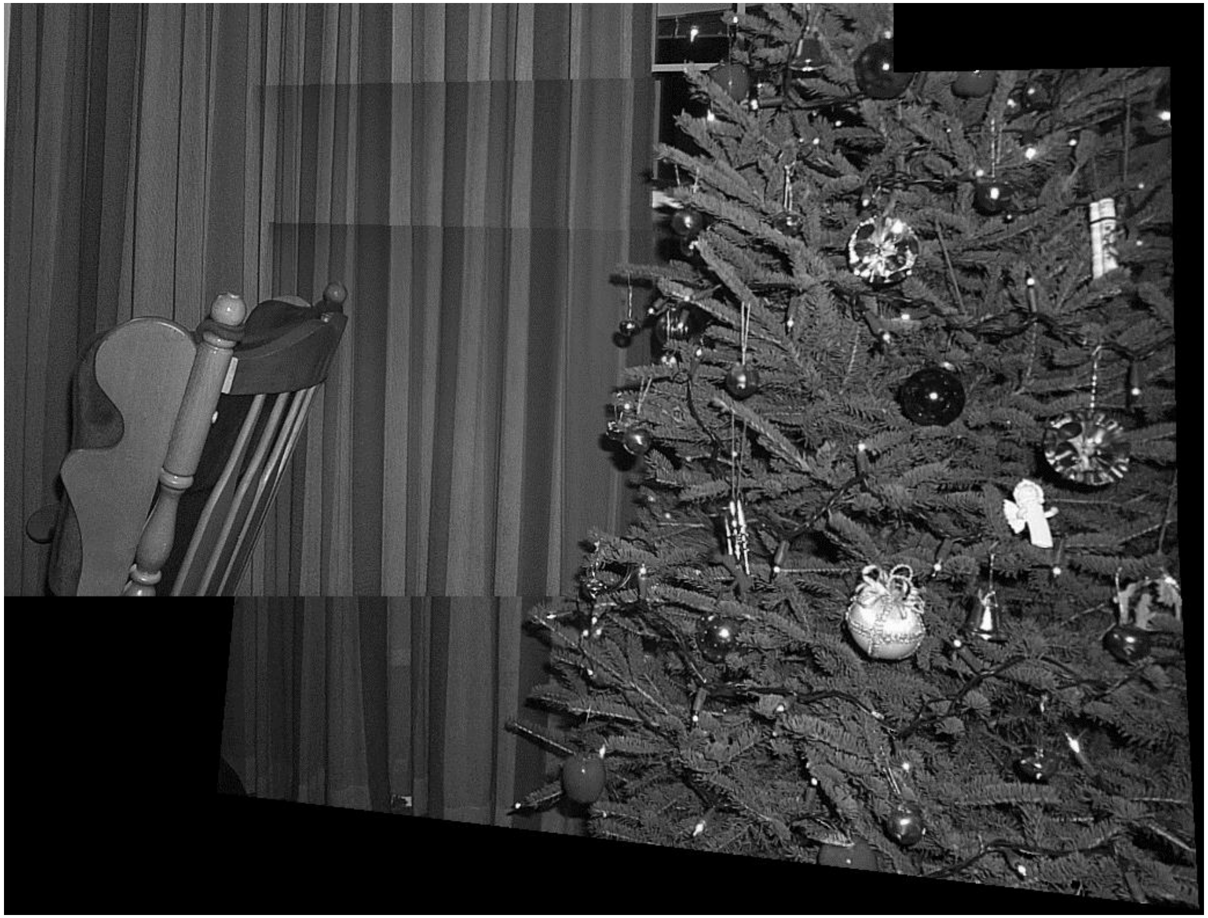
Automatic Mosaicing - Output



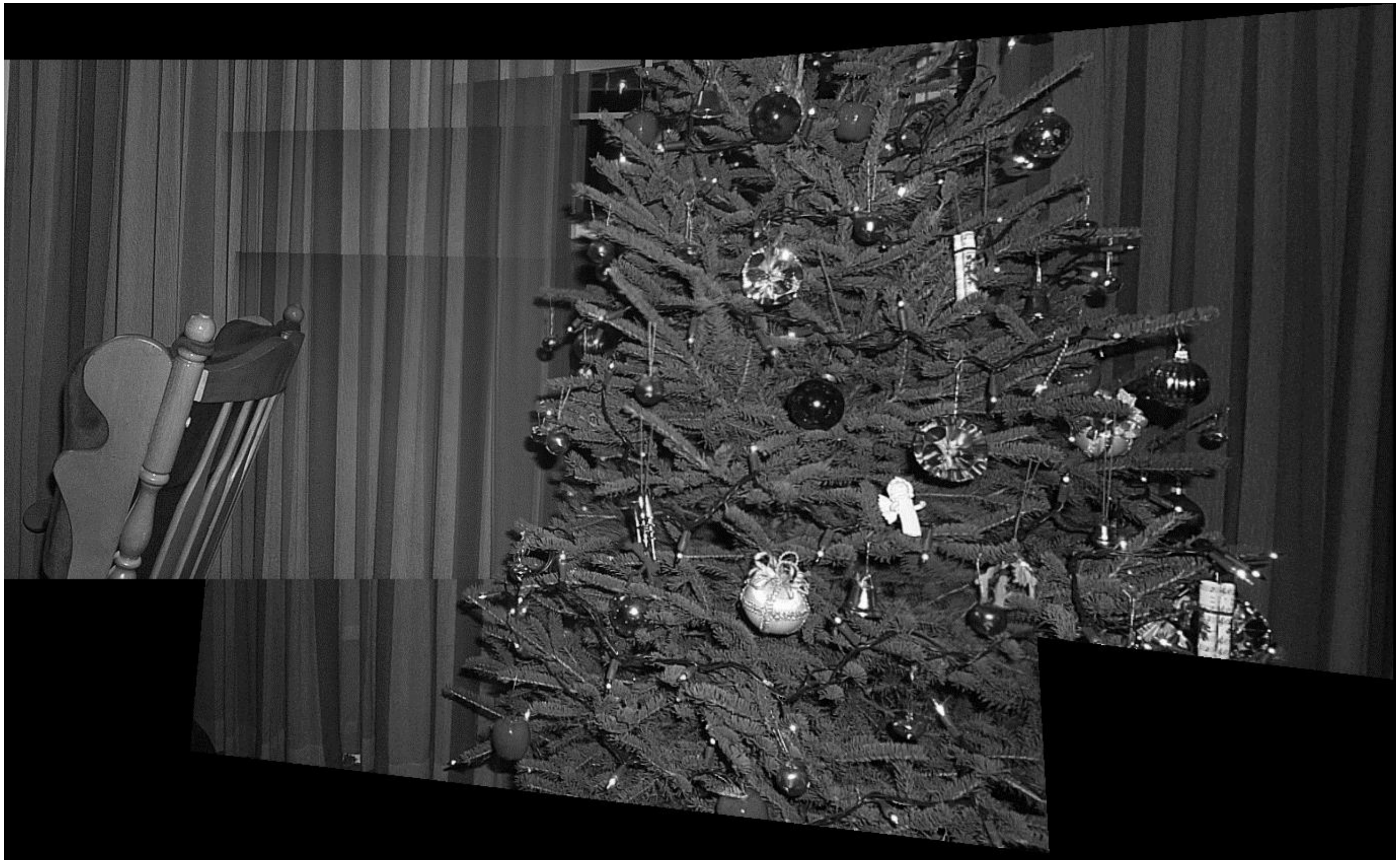
Automatic Mosaicing - Output



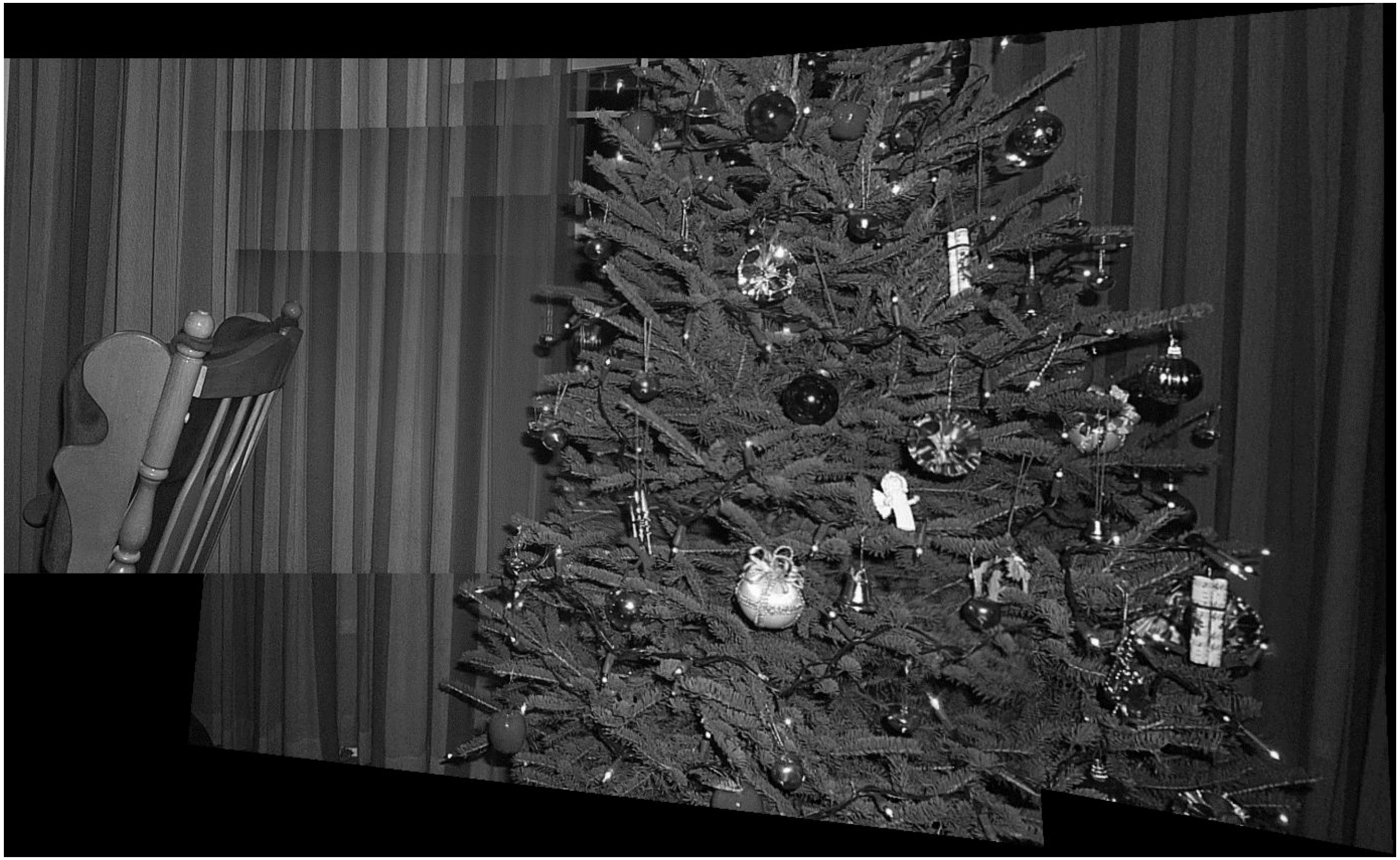
Automatic Mosaicing - Output



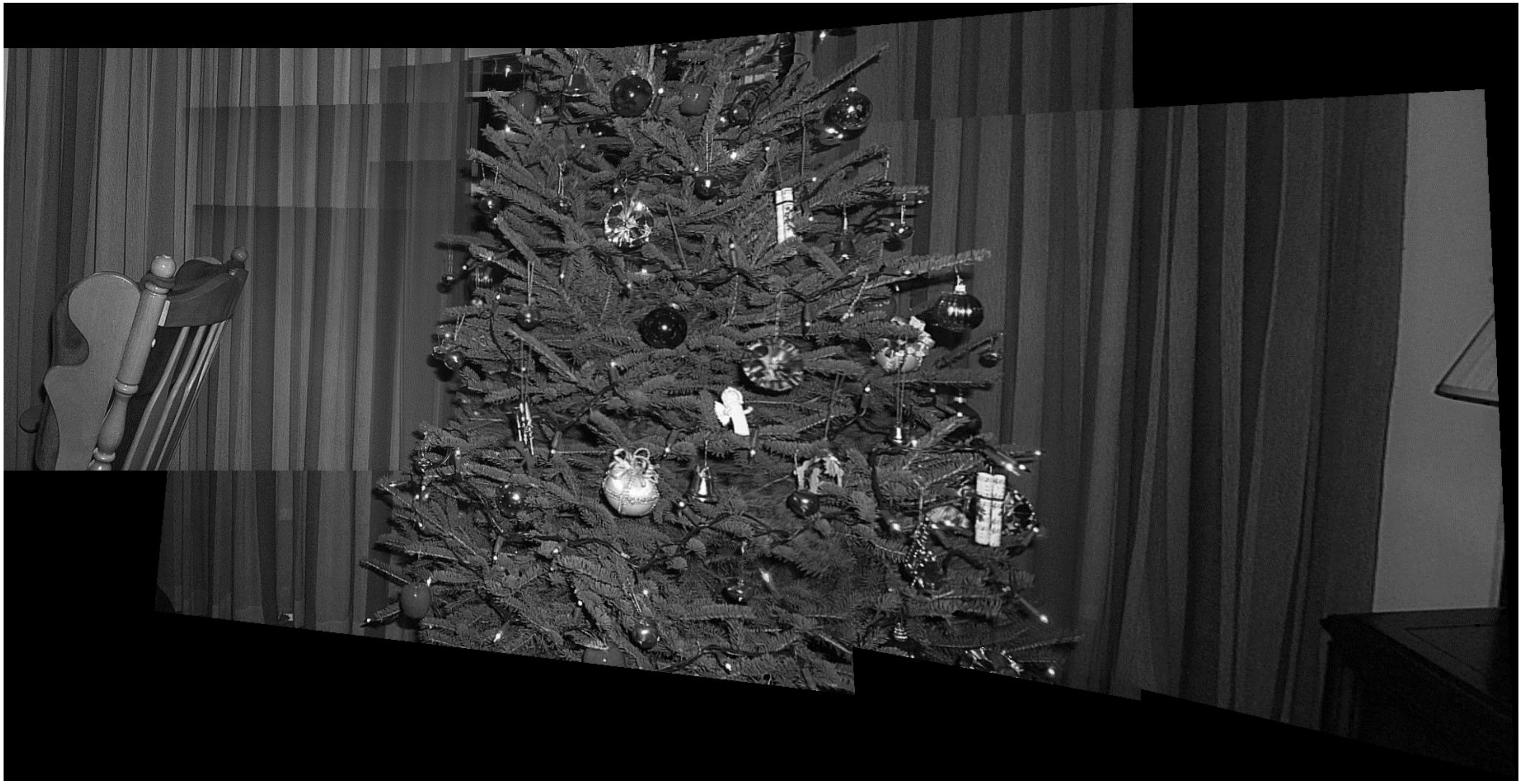
Automatic Mosaicing - Output



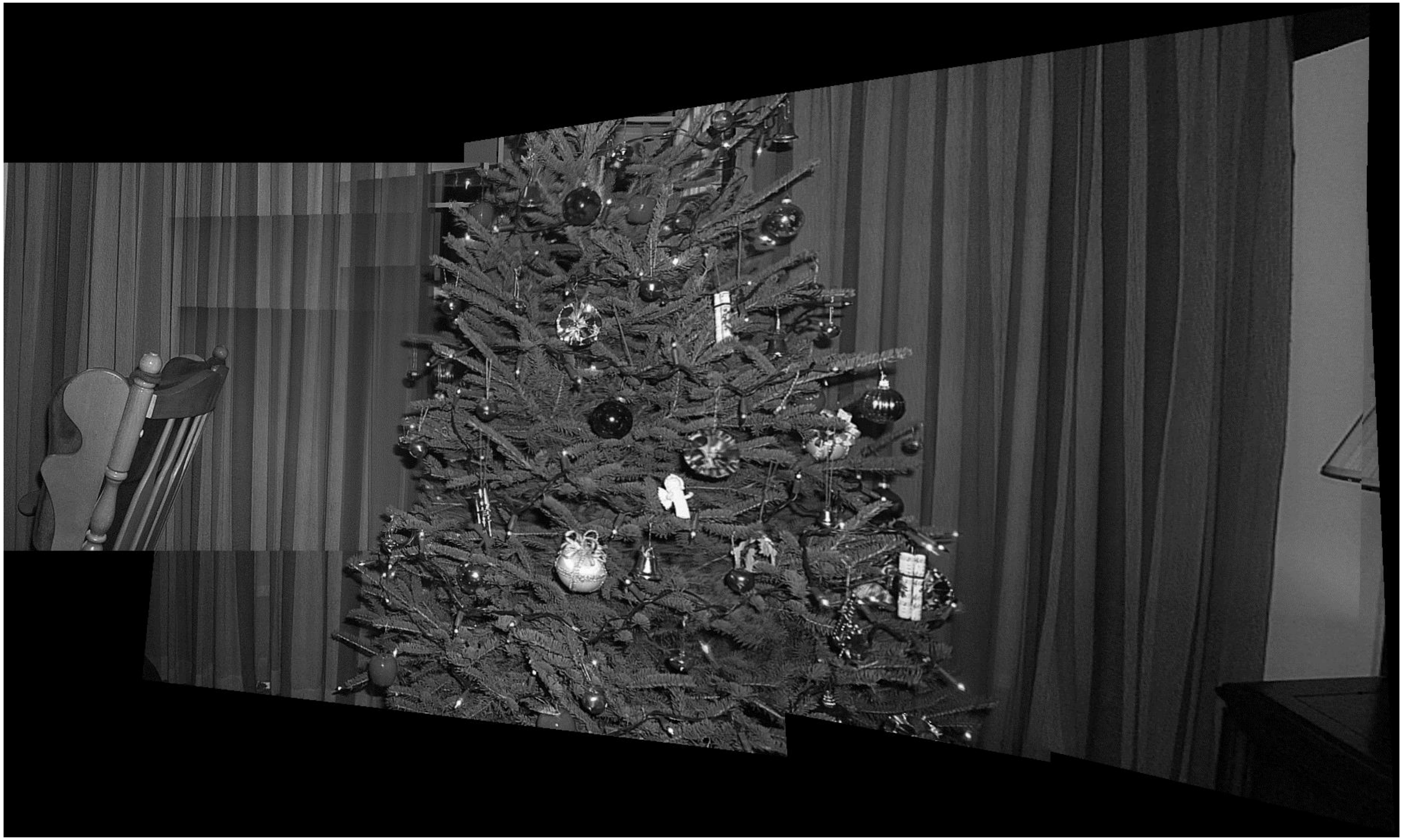
Automatic Mosaicing - Output



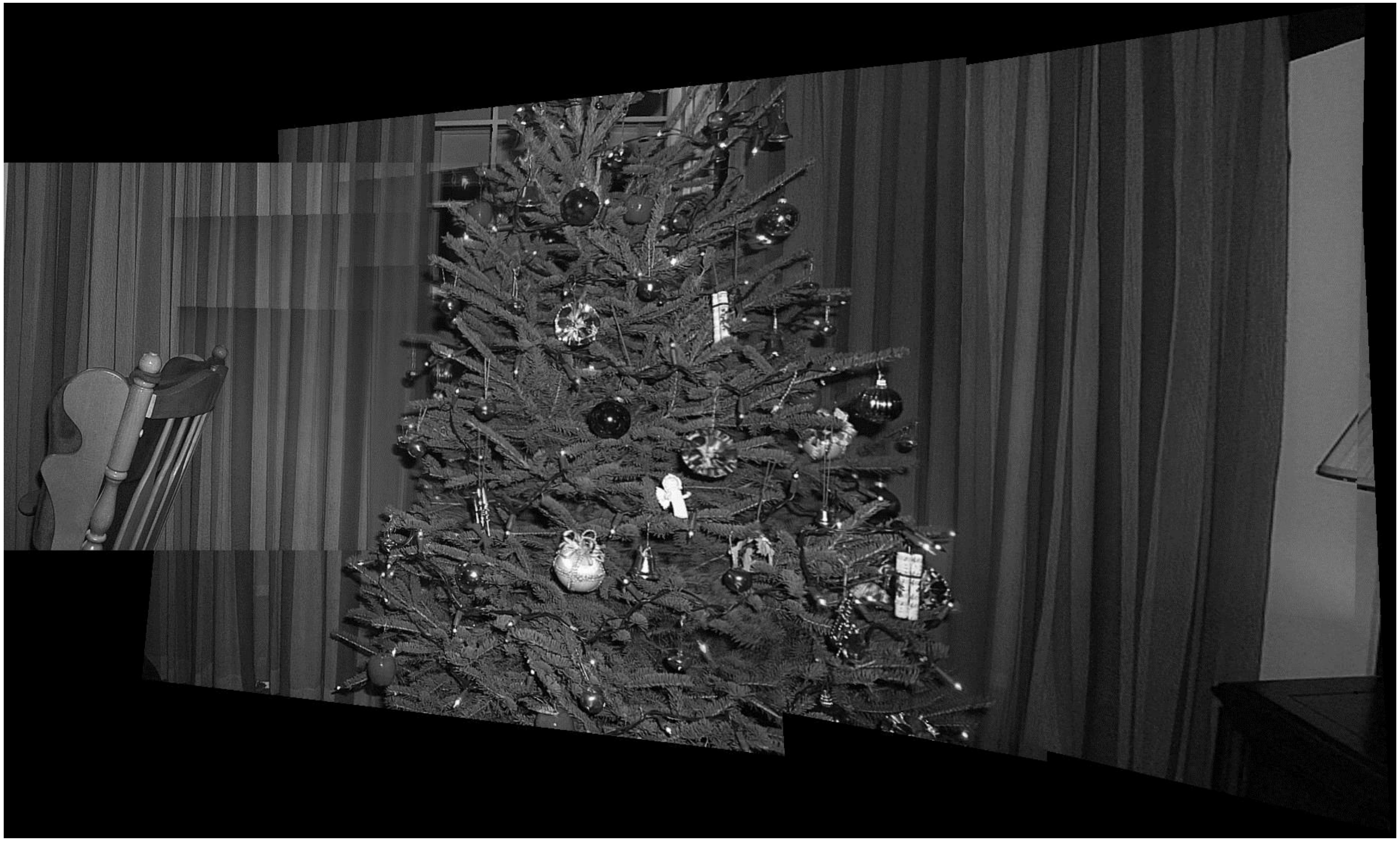
Automatic Mosaicing - Output



Automatic Mosaicing - Output



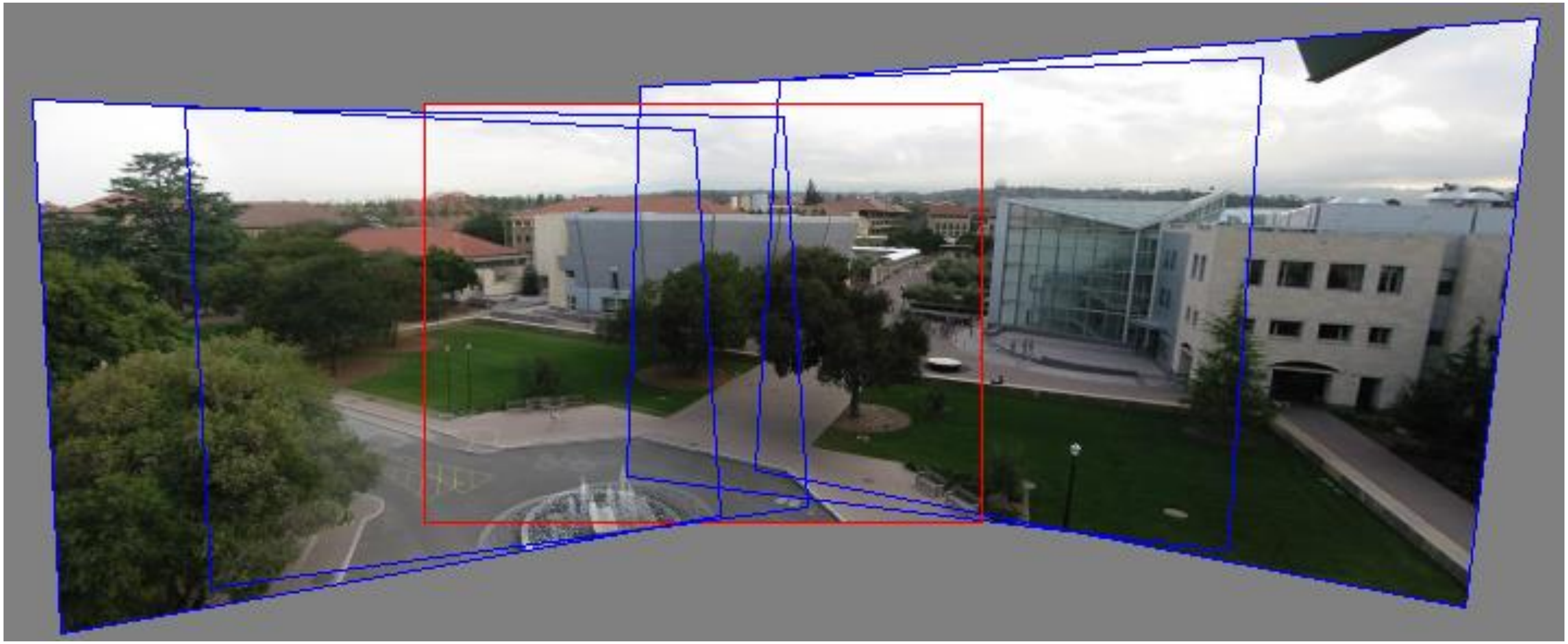
Automatic Mosaicing – Output



Mosaics: stitching images together

virtual wide-angle camera

Mosaics

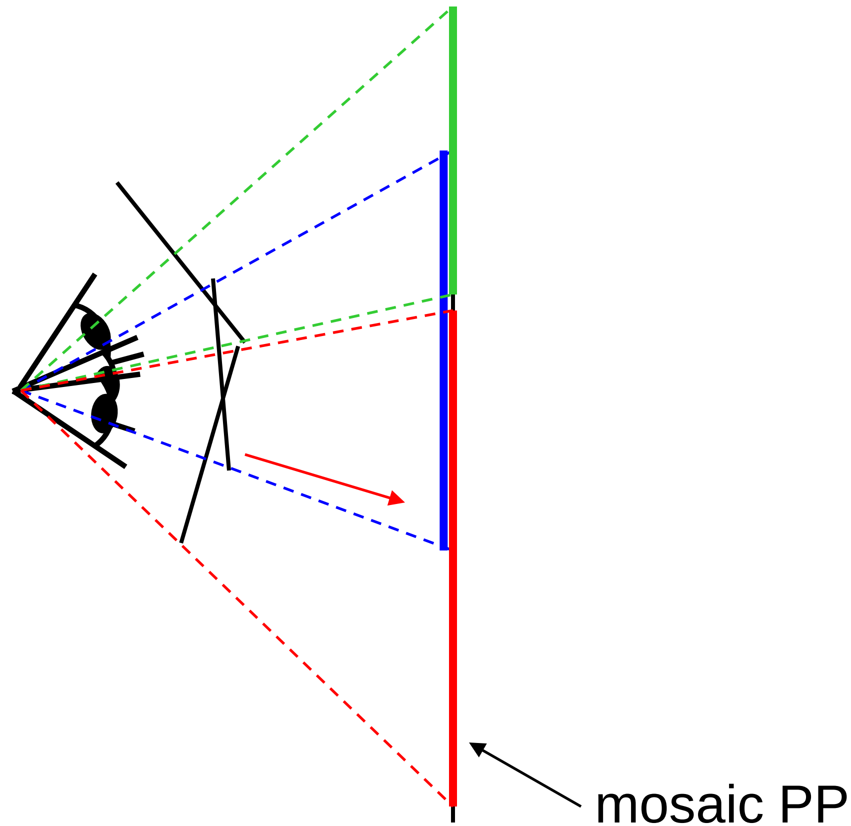


1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

Computing a mosaic from an image set

- Take a camera on a tripod
  - Rotate it around the axis of projection
- Choose one of the images as the base image
  - Compute the homography that aligns all the other images relative to that image
- You get a single large image, a mosaic which looks like a high resolution image taken from that viewpoint
  - This large image is called a mosaic
- Sometimes mosaics are called panoramas if they are wide enough (360 degrees or close)
  - Both names (mosaic/panorama) are used

Image reprojection



The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Computing special mosaics

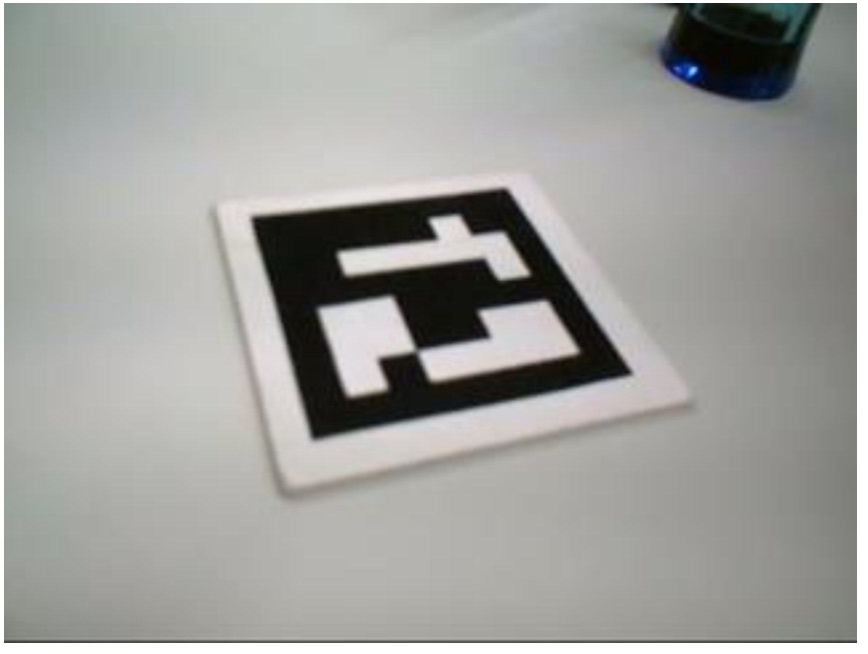
- You can synthesize any view (not a fixed reference view) to make a virtual camera
  - Need special real-time viewer to take mosaic and synthesize any rotated view
  - Use a speical re-sampling algorithm to create a new view
- Quicktime VR is a panorama creation/viewing system that can do this (many others)
  - Very commonly used in many practical applications
- Microsoft ICE = automatic panorama software
  - Rotate camera through any number of views
  - Give system views in any order (and views may be zoomed)
  - As long as have sufficient overlap, panorama will be created
  - Can even create High Definition panoramas!

Augmented Reality

- Print self encoding binary pattern (tags)
- In real-time AR software
  - Recognizes the tag, and finds pixel locations of 4 corners
  - Computes homography that maps these four corner pixels to a rectified front facing image
  - Uses this homography to compute the camera pose relative to the tag in the real world
  - Draws a virtual oboject on top of the tag in the world
- Virtual content can be any 3d model, etc.
- Very common systems in practice and available on cell phones
  - ARToolKit and ARTag

Typical binary tag pattern

- Bit string with error correction/detection is used to create a tag ID
- Predefined association of 3d models for each different tag ID



Using Tags for Augmentation

- In real-time find camera pose relative to the tag, and use this to draw virtual objects
- AR tags get an ID and camera position!

Homography

- Very useful in practice because of mosaics!
- Rectification is also important and will be used in stereo vision and other applications
- Many common mosaicing systems exist
- But math still depends on basic homography
  - Many other issues to create an automatic mosaicing system
  - Need to extract features (interest points!)
  - Match them between images
  - Align all the images with homographies
  - Then blend them all together properly