Planar homographies

Goal: Introducing 2D Homographies

Motivation:

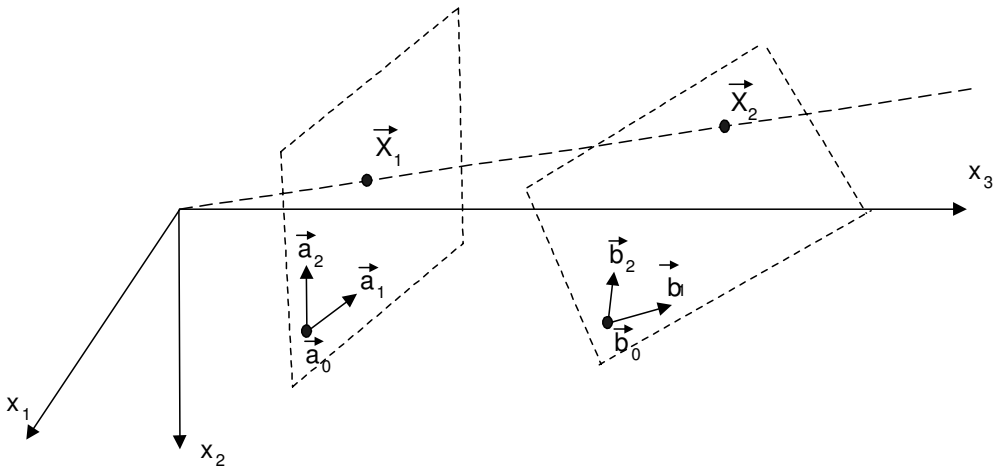
- What is the relation between a plane in the world and a perspective image of it?
- Can we reconstruct another view from one image?

Readings:

- http://www.robots.ox.ac.uk/ vgg/projects/SingleView/
- Philip H. S. Torr and A. Zisserman, *Feature Based Methods for Structure and Motion Estimation*,

Matlab Tutorials: warpDemo.m

Definition



- Let us consider a 3D coordinate frame and two arbitrary planes
- The first plane is defined by the point $\vec{b_0}$ and two linearly independent vectors $\vec{b_1}$, $\vec{b_2}$ contained in the plane
- Let us consider a point $\vec{X_2}$ in this plane. Since the vectors $\vec{b_1}$ and $\vec{b_2}$ form a basis in this plane, we can express $\vec{X_2}$ as:

$$\vec{X_2} = q_1\vec{b_1} + q_2\vec{b_2} + \vec{b_0} = \left(\vec{b_1}, \vec{b_2}, \vec{b_0}\right) \begin{pmatrix} q_1 \\ q_2 \\ 1 \end{pmatrix} = B\vec{q}$$

  where:

  - $B = \left(\vec{b_1}, \vec{b_2}, \vec{b_0}\right) \in \Re^{3X3}$ defines the plane
  - $\vec{q} = (q_1, q_2, 1)^T$ defines the 2D coordinates of $\vec{X_2}$ with respect to the basis $\left(\vec{b_1}, \vec{b_2}\right)$

Definition

- We can write a similar identity for the second plane

$$\vec{X_1} = A\vec{p}$$

  where:

  - $A = (\vec{a_1}, \vec{a_2}, \vec{a_0}) \in \Re^{3X3}$ defines the plane
  - $\vec{p} = (p_1, p_2, 1)^T$ defines the 2D coordinates of $\vec{X_2}$ with respect to the basis $(\vec{p_1}, \vec{p_2})$

- We now impose the constraint that point $\vec{X_1}$ maps to point $\vec{X_2}$ under perspective projection, centered at the origin $\vec{X} = \vec{0}$, so

$$\vec{X_1} = \alpha(\vec{q})\vec{X_2},$$

  where:

  - $\alpha(\vec{q})$ is a scalar that depends on $\vec{X_2}$, and consequently on $\vec{q}$

- By combining the equation above with the constraint that each of the two points must be situated in its corresponding plane, one obtains the relationship between the 2D coordinates of these points:

$$\vec{p} = \alpha(\vec{q})A^{-1}B\vec{q}$$

Definition

- Note that the matrix A is invertible because $\vec{a_0}$,$\vec{a_1}$,$\vec{a_2}$ are linearly independent and nonzero (the two planes do not pass through origin)
- Also note that the two vectors $\vec{p}$ and $\vec{q}$ above have a unit third coordinate
- Hence the role of $\alpha(\vec{q})$ is simply to scale the term $\alpha(\vec{q})A^{-1}B\vec{q}$ such that its third coordinate is 1
- We can get rid of this nonlinearity by moving to homogeneous coordinates:

$$\vec{p^h} = H\vec{q^h} \tag{1}$$

  where:

  – $\vec{p^h}$, $\vec{q^h}$ are homogeneous 3D vectors

  – $H \in \Re^{3X3}$ is called a **homography matrix** and has 8 degrees of freedom, because it is defined up to a scaling factor ($H = cA^{-1}B$ where c is any arbitrary scalar)

- The mapping defined by (1) is called a 2D **homography**

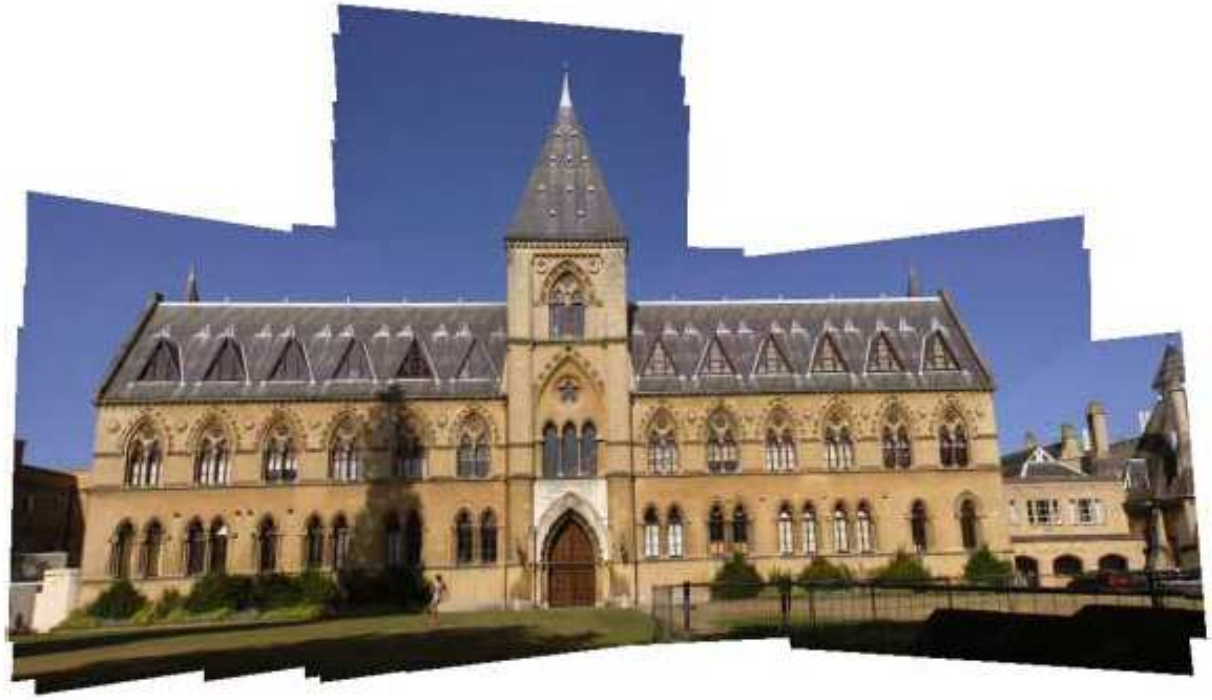
Applications of Homographies

- Here are some computer vision and graphics applications that employ homographies:
  - mosaics (image processing)
    - Involves computing homographies between pairs of input images
    - Employs **image-image** mappings
  - removing perspective distortion (computer vision)
    - Requires computing homographies between an image and scene surfaces
    - Employs **image-scene** mappings
  - rendering textures (computer graphics)
    - Requires applying homographies between a planar scene surface and the image plane, having the camera as the center of projection
    - Employs **scene-image** mappings
  - computing planar shadows (computer graphics)
    - Requires applying homographies between two surfaces inside a 3D scene, having the light source as the center of projection
    - Employs **scene-scene** mappings

Recovering Homographies between images: Mosaics

- Assume we have two images of the same scene from the same position but different camera angles
- It is easy to show that the mapping between the two image planes is also a homography, independently of the structure (depth) of the scene
- We can look for a set of points in the left image and find the corresponding points in the right image based on image features
- Since the homography matrix H has 8 degrees of freedom, 4 corresponding $(\vec{p}, \vec{q})$ pairs are enough to constrain the problem
- Application: mosaics - building a wide angle image by stitching together several images taken under different orientations from the same position

Mosaics (example)



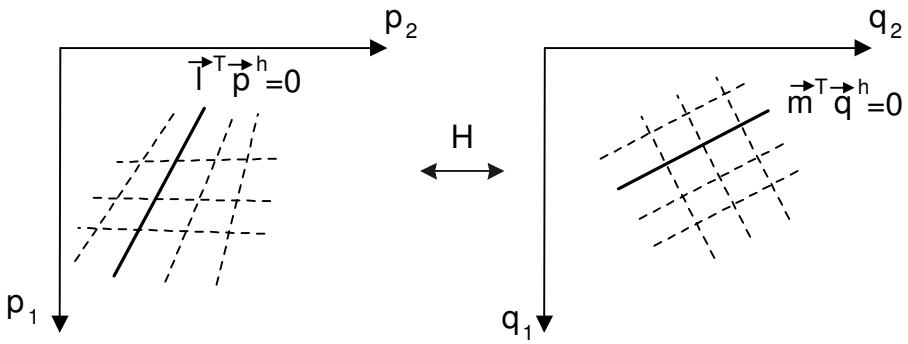
What happens when camera position changes?

- But what if camera position also changes? It is easy to see that in the general case the transformation between two such images is no longer a homography
- However, for image points corresponding to the same planar surface, the image-image transformation remains a homography
- Hence in this case, different homographies exist between subregions of the two images that correspond to the same planar surfaces
- More on this later, since this will be the subject of a later assignment

Vision: Removing perspective distortion

- Assume you are given a single image, containing one or more planar surfaces
- Can we recover the texture from the planar surface(s)?
- This is a hard problem, because not enough information is available (sound familiar?)
- This is a simplified version of the problem of recovering intrinsic images (we are trying to separate albedo from depth information)
- Likewise, there are two ways to deal with this:
  - Make some assumptions about the world
  - Cheat and use humans to communicate their assumptions to the machine
- Here are two approaches that use the last of the choices above
  - Let the user specify the **horizon line**
  - Let the user specify sets of lines in each plane that he thinks are **parallel** in the real world (from which the horizon can be determined)
- Building a system that makes such assumptions on its own is still an open research problem

Mapping lines from texture space to image space



- Let us consider a homography from texture coordinates to image coordinates:

$$\vec{p^h} = H\vec{q^h}$$

- Take an arbitrary line in texture space given by the equation:

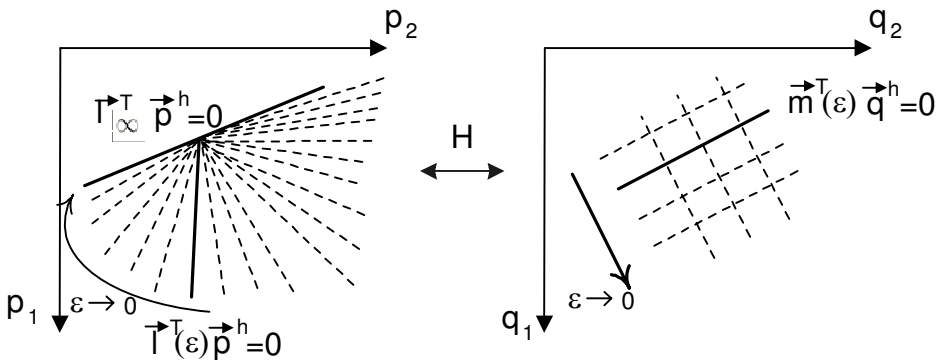
$$\vec{m^T}\vec{q^h} = 0$$

- Then the corresponding line in image coordinates is given by:

$$\vec{l^T}\vec{p^h} = 0$$

where $\vec{l} = \left(H^{-1}\right)^T \vec{m}$

Using horizon line to remove distortion.



- Let us consider the sequence of lines in texture space:

$$\vec{m}^T(\epsilon) \cdot \vec{q}^h = 0, \quad \text{where } \vec{m}(\epsilon) = \begin{pmatrix} \epsilon\vec{n} \\ 1 \end{pmatrix},$$

  with $\vec{n}$ is an arbitrary unit vector.

- This sequence of lines converges to infinity in texture space, since the point on the line closest to the origin is $\vec{q}_0$:

$$\vec{q}_0 = -\frac{1}{\epsilon}\vec{n}, \ \lim_{\epsilon \to 0} |\vec{q}_0| = \infty.$$

  The parameter vector for the line converges to

$$\vec{m}_\infty = \lim_{\epsilon \to 0} \vec{m}(\epsilon) = \begin{pmatrix} \vec{0} \\ 1 \end{pmatrix}$$

- In the image space, this line corresponds to the horizon line $\vec{l_\infty}$:

$$H^T \vec{l_\infty} = \begin{pmatrix} \vec{0} \\ 1 \end{pmatrix}$$

Using horizon line to remove distortion.

Suppose we know the horizon line parameters $\vec{l_\infty} = c(\vec{n}^T, a)^T$, where c is an arbitray constant and $a > 0$. What does this tell us about the homography H between the scene coordinates and the image coordinates?

The previous equation, $H^T \vec{l_\infty} = (\vec{0}^T, 1)^T$, has one particular solution given by

$$H = H_p = \begin{pmatrix} aI & \vec{n} \\ -\vec{n}^T & a \end{pmatrix}$$

Other solutions can be written in the form $H = H_p H_a$ where

$$H_a^T \begin{pmatrix} \vec{0} \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{0} \\ 1 \end{pmatrix}$$

This implies

$$H_a = \begin{pmatrix} A & \vec{b} \\ 0 & 1 \end{pmatrix}$$

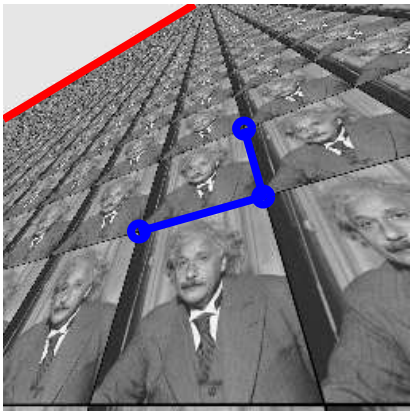
where $A \in \Re^{2X2}$ is any nonsingular matrix and $\vec{b}$ is any 2D vector.

- H_a is actually any affine transformation
- It can be shown that the inverse of the matrix H_a is also affine

Using horizon line to remove distortion (example)

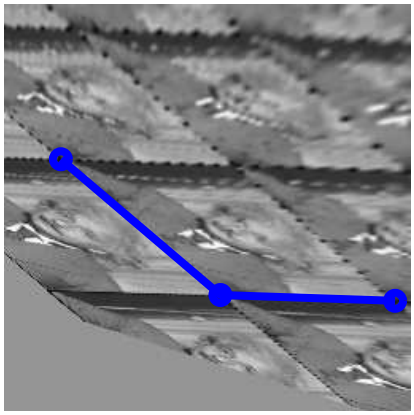
- Thus we can determine the mapping of the texture from scene to image up to an affine transformation just by knowing the horizon line. To go further, we need additional constraints from the user.
- Usually, these come as a pair of vectors in image space which we assume to be unit length and perpendicular in texture space

Original



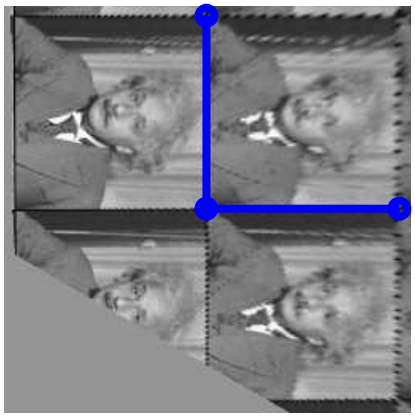
Range: [0, 226]
Dims: [256, 256]

Affine Reconstruction



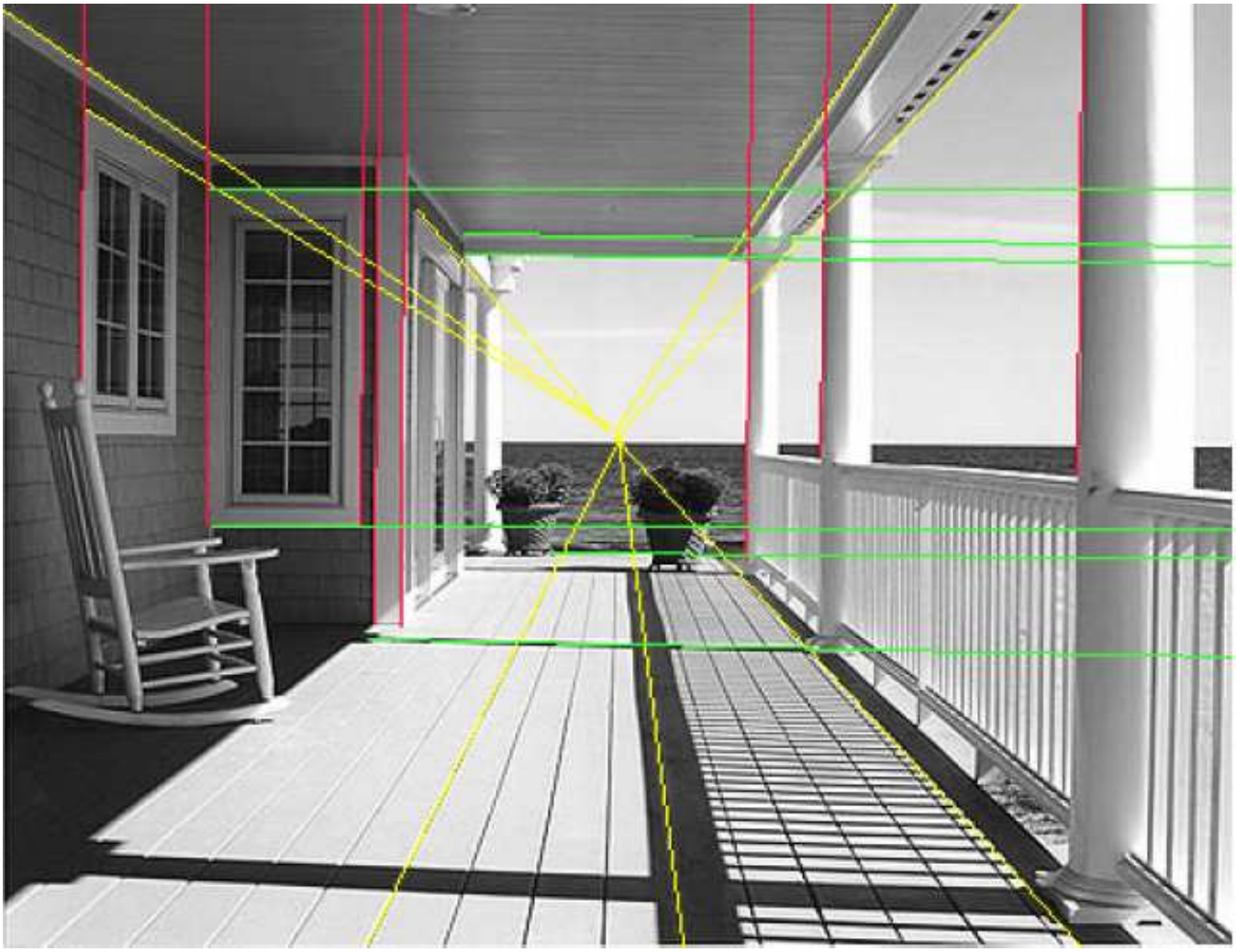
Range: [0, 219]
Dims: [256, 256]

Similarity Reconstruction



Range: [0.0378, 219]
Dims: [256, 256]

Using parallel lines to remove distortion and reconstruct scene structure (example)



Using parallel lines to remove distortion and reconstruct scene structure (example)