[The material in the first part of these notes was presented in lecture 16 class and slides.]

Homographies

Let's now look at another image transformation, called a *homography*, which arises very often in real scenes, both manmade and natural.

Case 1: scene plane to image pixels

Suppose we have a planar surface in the world (e.g. a wall, a ground plane) and we view it with a camera with projection matrix $\mathbf{P}$. The planar surface is 2D and so we can give it a coordinate system (s,t). Points on the plane are situated in the 3D world and their 3D positions can be expressed in XYZ *world coordinates*. The transformation from (s,t) coordinates to world coordinates XYZ can be obtained by multiplying $(s,t,1)$ by a 4×3 matrix, as follows:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & b_x & X_0 \\ a_y & b_y & Y_0 \\ a_z & b_z & Z_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix} \tag{1}$$

The first two columns can be interpreted as direction vectors corresponding to the coordinate system's basis vectors, namely where $(1,0,0)$ and $(0,1,0)$ are mapped to. The third column is the 3D world coordinate position of the *origin* of the plane, i.e. $(s,t) = (0,0)$. This mapping takes the origin $(s,t) = (0,0)$ to (X_0, Y_0, Z_0). It takes the corner $(s,t) = (0,1)$ to $(X_0 + b_x, Y_0 + b_y, Z_0 + b_z)$, and it takes the corner $(s,t) = (1,0)$ to $(X_0 + a_x, Y_0 + a_y, Z_0 + a_z)$, etc.

The image pixel (x,y) corresponding to a point (s,t) in the scene plane is obtained by projection:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} a_x & b_x & X_0 \\ a_y & b_y & Y_0 \\ a_z & b_z & Z_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

The combined mapping takes $(s,t,1)'$ to $(wx, wy, w)'$ defines a 3×3 matrix $\mathbf{H}$,

$$\mathbf{H} \equiv \mathbf{P} \begin{bmatrix} a_x & b_x & X_0 \\ a_y & b_y & Y_0 \\ a_z & b_z & Z_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In general, a 3×3 matrix $\mathbf{H}$ that is invertible is called a *homography*. The inverse of this mapping is:

$$\begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

The inverse is well-defined as long as the camera position does not belong to the 3D planar surface. To see why, recall $\mathbf{P} = \mathbf{K}[\mathbf{I}|-\mathbf{c}]$. Since $\mathbf{K}$ is defined to be invertible, the matrix $\mathbf{H}$ will fail to be invertible exactly when

$$[\ \mathbf{I}\ |\ -\mathbf{c}]\begin{bmatrix} a_x & b_x & X_0 \\ a_y & b_y & Y_0 \\ a_z & b_z & Z_0 \\ 0 & 0 & 1 \end{bmatrix}$$

fails to be invertible, or equivalently that the 3×3 matrix

$$\begin{bmatrix} a_x & b_x & X_0 - c_x \\ a_y & b_y & Y_0 - c_y \\ a_z & b_z & Z_0 - c_y \end{bmatrix}$$

is not invertible, which means that its columns are linearly dependent. This means in particular that the vector from the camera $\mathbf{c}$ to the 3D origin (X_0, Y_0, Z_0) of the plane is a linear combination of vectors $\mathbf{a}$ and $\mathbf{b}$. But this can only happen when the camera position lies in the scene plane in question.

Case 2 : two cameras, one scene plane

Suppose the same scene plane is viewed by a second camera, which would have a different $\mathbf{P}$ matrix. We would now have two homographies $\mathbf{H}_1$ and $\mathbf{H}_2$, defined by the two cameras. This implies that the composite mapping $\mathbf{H}_2\mathbf{H}_1^{-1}$ maps pixel coordinates in the first camera to pixel coordinates in the second camera. That is, each camera defines a homography of the form

$$\mathbf{H}^{-1}\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

but the right side is the same for both (since it is independent of the camera), so we just equate the left sides for the two cameras. Let (x, y) denote image coordinates for camera 1 and let (x', y') denote image coordinates for camera 2.

$$\mathbf{H}_2^{-1}\begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix} = \mathbf{H}_1^{-1}\begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

and so

$$\begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix} = \mathbf{H}_2\mathbf{H}_1^{-1}\begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

Note that this construction relies critically on the scene being a planar surface.

Case 3 : panoramas and image stitching

Homographies can arise for general (non-planar) scenes as well, namely if you have one camera and you use it to take more than one image by rotating the camera around the center of projection. This is indeed what we do to stitch images together to make a panorama.

To see that two images taken under this condition (of pure rotation) are related by a homography, let (X, Y, Z) be a scene point written in the scene coordinate system. Then the image positions in the two images will be

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \mathbf{K}\mathbf{R}_1[\mathbf{I} \mid -\mathbf{c}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

and

$$\begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix} = \mathbf{K}\mathbf{R}_2[\mathbf{I} \mid -\mathbf{c}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

for the two orientations $\mathbf{R}_1$ and $\mathbf{R}_2$ of the camera. Then,

$$(\mathbf{K}\mathbf{R}_1)^{-1} \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = (\mathbf{K}\mathbf{R}_2)^{-1} \begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix}$$

and so

$$\mathbf{K}\mathbf{R}_2(\mathbf{K}\mathbf{R}_1)^{-1} \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} w'x' \\ w'y' \\ w' \end{bmatrix}$$

Thus, we have a homography $\mathbf{K}\mathbf{R}_2(\mathbf{K}\mathbf{R}_1)^{-1}$ that takes pixel coordinates (x, y) in camera orientation 1 to pixel coordinates (x', y') in camera orientation 2.

Notice that the distances from the camera to the scene point, $\| (X, Y, Z)^T - \mathbf{c} \|$, plays no role in the homography. If you are only considering what the scene looks like from some point (the center of projection of the camera) then the scene points could be at any distance (even infinity) along the rays that arrive at the camera.

Case 4: Image rectification

In an upcoming lecture, we will discuss the binocular stereo problem. Many solutions to this problem require that one first "rectifies" two images. As we will see, rectification involves homographies, so since we are considering homographies here, let's look at rectification. We will later show how they are used in stereovision.

For now, let's suppose that we know the internals and externals of *two* cameras, $\mathbf{K}_1$, $\mathbf{K}_2, \mathbf{R}_1$, $\mathbf{R}_2, \mathbf{c}_1, \mathbf{c}_2$. Define $\mathbf{T} = \mathbf{c}_2 - \mathbf{c}_1$ to be the vector from camera 1 to camera 2. We can write $\mathbf{T}$ in any basis we like. In the argument that follows, we write it in the basis of camera 1, e.g. if $\mathbf{T}$ were $(1, 0, 0)^T$ then $\mathbf{T}$ would be in the camera 1's X axis direction. The reason that we write $\mathbf{T}$ in camera 1's coordinates is that we are designing a rotation matrix to be applied to points in camera 1's coordinate system.

Let $\hat{\mathbf{z}}_1 = (0, 0, 1)$ be camera 1's optical axis direction. Define the following rotation matrix that would rotate camera 1 so that the vector $\mathbf{T}$ would become the new X axis direction for camera 1,

$$\mathbf{R}_{rect} = \begin{bmatrix} unit(\mathbf{T}) \\ unit(\hat{\mathbf{z}}_1 \times \mathbf{T}) \\ unit(\mathbf{T} \times (\hat{\mathbf{z}}_1 \times \mathbf{T})) \end{bmatrix}.$$

where $unit(\cdot)$ is the operator that divides a vector by its magnitude. By inspection, the rows of this matrix are orthonormal and

$$\begin{bmatrix} |\mathbf{T}| \\ 0 \\ 0 \end{bmatrix} = \mathbf{R}_{rect}\mathbf{T}.$$

You can also confirm that this coordinate system is left-handed ($\mathbf{x} \times \mathbf{y} = \mathbf{z}$) by noting that the three rows will be the **x,y,z** axes of the rotated camera. We've defined the new $\mathbf{x}$ axis to be in the $\mathbf{T}$ direction, and so the second row says that the $\mathbf{y}$ axis will be $\mathbf{z} \times \mathbf{x}$, which is correct for the usual handedness rules. The third row gives the new $\mathbf{z}$ axis and again obeys the correct handness rule, namely $\mathbf{x} \times \mathbf{y} = \mathbf{z}$.

We would also like to know how to rotate camera 2 so that its orientation becomes the same as the orientation of (the rotated) camera 1. Since the matrix $\mathbf{R}_1\ \mathbf{R}_2^{-1}$ maps from camera 2's coordinate system to camera 1's coordinate system[1], we apply the rotation $\mathbf{R}_{rect}\mathbf{R}_1\ \mathbf{R}_2^{-1}$. We now have rotations that would bring the cameras into an orientation where their XYZ axes are parallel and the X axis is in the direction from camera 1 to camera 2.

In the slides, I referred to the real cameras and "virtual" cameras. We are not actually going to rotate the real cameras. Rather we are going to apply homographies to the image which have a similar effect as if we rotated the cameras. I say similar rather than identical. A key difference here is that if we actually rotate a camera, then we bring new parts of the scene into view and we lose other parts of the scene from view. However, when we virtually rotate the cameras, we don't change what each camera sees. Rather we just deform it. The deformation is a homography.

In addition to performing a virtual rotation, we will re-map the pixels in the two cameras so that the camera internals will be the same. Specifically, we will use the camera internal matrix of camera 1.

For any pixel (x, y) in the original camera 1 image, we undo the camera projection matrix to write the point in mm instead of in pixels, then perform the rotation $\mathbf{R}_{rect}$, and then remap to pixel coordinates.

$$\begin{bmatrix} w\tilde{x}_1 \\ w\tilde{y}_1 \\ w \end{bmatrix} = \mathbf{K}_1\mathbf{R}_{rect}\mathbf{K}_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

For a pixel in camera 2, we apply a slightly different mapping which will take us to a pixel representation as if camera 2's internals were the same as camera 1's.

$$\begin{bmatrix} w\tilde{x}_2 \\ w\tilde{y}_2 \\ w \end{bmatrix} = \mathbf{K}_1\mathbf{R}_{rect}\mathbf{R}_1\mathbf{R}_2^{-1}\mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

The result is a pair of images that have been rectified so that have pixels that correspond to two versions of the same camera, separated by a spatial translation of distance $|T|$ in the cameras' (now shared) X axis direction. We will return to this configuration next week, when we discuss the stereo problem.

[1] more precisely, it maps from camera 2's coordinate system to world coordinates and from world coordinates to camera 1's coordinate system

Solving for the homography between two images

We next discuss a general problem of how to find a homography that maps from one image to the other. We first assume that we are given N sets of corresponding point pairs (x_i, y_i) and $(x_i',\ y_i')$ between two images which are *approximately* related by an *unknown* homography $\mathbf{H}$, namely

$$\begin{bmatrix} w'x_i' \\ w'y_i' \\ w' \end{bmatrix} \approx \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

The points might be image features such as Harris corners or SIFT features. (We will discuss later how to find these corresponding points.) Note that we don't expect to localize these points exactly - there may be errors.

We want to solve for $\mathbf{H}$. As we did with camera calibration, we can rewrite the above approximation as:

$$x_i' = \frac{w_i x_i'}{w_i} \approx \frac{H_{11}x_i + H_{12}y_i + H_{13}}{H_{31}x_i + H_{32}y_i + H_{33}}$$
$$y_i' = \frac{w_i y_i'}{w_i} \approx \frac{H_{21}x_i + H_{22}y_i + H_{23}}{H_{31}x_i + H_{32}y_i + H_{33}}$$

and so

$$x_i'(H_{31}x_i + H_{32}y_i + H_{33}) \approx H_{11}x_i + H_{12}y_i + H_{13}$$
$$y_i'(H_{31}x_i + H_{32}y_i + H_{33}) \approx H_{21}x_i + H_{22}y_i + H_{23}.$$

which we can rewrite as:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ : & : & : & : & : & : & : & : & : \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_N'x_N & -x_N'y_N & -x_N' \\ 0 & 0 & 0 & x_N & y_N & 1 & -y_N'x_N & -y_N'y_N & -y_N' \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ : \\ 0 \\ 0 \end{bmatrix}$$

The least squares solution for $\mathbf{H}$ is obtained by taking the $2N \times 9$ data matrix $\mathbf{A}$ and find the eigenvector of $\mathbf{A}^T\mathbf{A}$ having smallest eigenvalue.

A few important points are worth noting. First, can we relate this problem to the camera calibration problem, in terms of a particular geometric configuration? Yes. A calibration grid such a checkerboard pattern on the faces of a cube is a set of points (X_i, Y_i, Z_i). If all points lie on a single plane, we can think of the plane as $Z_i = 0$ (in a suitable world coordinate system) for all i, and $(s, t) \equiv (X_i, Y_i)$. Since $Z_i = 0$ for all i, it plays no role in the problem and this drops the number of constraints from 12 to 9, namely it drops the third column of the $\mathbf{P}$ matrix, giving essentially the problem above where $\mathbf{P}_{ij}$ are replaced by $\mathbf{H}_{ij}$.

Second, just as with the camera problem, it turns out we get better estimates if we normalize the data. Intuitively, the issue in the matrix above is that the values of x_i, y_i, x_i', y_i' can go from 1

to several hundred (pixel indices), and so terms in the matrix that involve their products can take values in 1 to order 10^5. This means that the matrix columns will each contain numbers with quite different ranges of values. Since the values may contain noise, this can lead to numerical problems and poor estimates of $\mathbf{H}$. Normalizing brings all values within the same range, which reduces these numerical problems.

To normalize, we subtract the means and divide by the standard deviations, similarly to what we defined last lecture.

$$\mathbf{M_1} : (x_i, y_i) \rightarrow (\frac{x_i - \bar{x}}{\sigma_1}, \frac{y_i - \bar{y}}{\sigma_1})$$

$$\mathbf{M_2} : (x_i', y_i') \rightarrow (\frac{x_i' - \bar{x'}}{\sigma_2}, \frac{y_i' - \bar{x'}}{\sigma_2}).$$

Then we solve for a homography $\mathbf{H}_{normalized}$ using least squares as above, but now we are using the normalized data points:

$$w_i \mathbf{M}_2 \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \approx \mathbf{H}_{normalized} \mathbf{M}_1 \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

If you are comparing the lecture notes to last lecture, then you'll note that today I am not using the "tilde" notation for the normalized coordinates.

Left multiplying by $(\mathbf{M}_2)^{-1}$, we see that the homography that we want to map the (x_i, y_i) space to the (x', y') space is:

$$\mathbf{H} = (\mathbf{M}_2)^{-1} \mathbf{H}_{normalized} \mathbf{M}_1 \ .$$

Using RANSAC

If we can be sure that our pairs of matching points can be modelled using a single homography then we would like to use as many pairs of matching points as possible. However, we rarely can be sure that the matches are all correct. As we discuss earlier in the course (lecture 6), we would like our method to deal with possibly wrong matches.

In the RANSAC method, we choose the *minimal* number of samples that we need to compute a model. In the case of fitting a homography, the minimal number of pairs is $N = 4$. If we have $N = 4$ pairs of corresponding 2D points $(x_i, y_i, x_i', y_i'), i = 1, \ldots, 4$ then we get a system of 8 equations with 9 unknowns. We can write $\mathbf{Ax=0}$ where $\mathbf{A}$ is an 8×9 matrix (see matrix on previous page) and note that we now insist on equality since we are solving exactly. A homography $\mathbf{H}$ that puts the four pairs of points into *exact* correspondence can be obtained by finding the null space of this $\mathbf{A}$ matrix. That is, $\mathbf{A}$ has 9 column vectors each with 8 elements. These column vectors cannot be linearly independent and so there must be a linear combination of them (the H_{ij}) that sums to the zero vector. We find $\mathbf{H}$ by solving for the null space of the 8×9 matrix $\mathbf{A}$. We can do this by taking the singular value decomposition $\mathbf{U\Sigma V^T}$ and taking the column vector $\mathbf{v}$ of $\mathbf{V}$ that corresponds to a singular value of 0.

Recall also that homographies only have eight degrees of freedom, not nine, since they operate on vectors that are in homogeneous coordinates. So we can multiply all elements in a homography matrix by the same value and we get the same transformation. Is this a problem? No. The solution space is one dimensional, namely a 1D null space. This dimension is corresponds exactly to the reduction by one degree of freedom in the homography.

Here is the typical RANSAC approach:

1. Find many feature points (SIFT features) in each of the two images: $\{(x_i, y_i, \dots)\}$ and $\{(x'_k, y'_k, \dots)\}$. For each feature point (x_i, y_i) in one image, find a set of candidate corresponding feature points in the second image (x'_i, y'_i) e.g. use SIFT descriptors. This give a set of 4-tuples (x_i, y_i, x'_i, y'_i). Note I am indexing now by i. This requires renumbering the feature points, so that corresponding features have the same index.

2. Randomly choose four 4-tuples from the set and fit an exact homography $\mathbf{H}$ that maps the four $\{(x_i, y_i)\}$ exactly to their corresponding $\{(x'_i, y'_i)\}$. Find the consensus set for that homography, namely find the number of other 4-tuples for whom the *distance* of the 4-tuple from the model is sufficiently small.

   The distance could be defined in a variety of ways, e.g. let $(x, y,\ x', y')$ be a new 4-tuple and we test whether it fits a homography $\mathbf{H}$ by computing

$$\begin{bmatrix} w\hat{x} \\ w\hat{y} \\ w \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

   and defining the distance/fit to be

$$dist(x, y, x', y'\ ;\ \mathbf{H}) \ = \ \| \ (\hat{x}, \hat{y}) - (x', y') \ \|_2 \ .$$

   If we are using SIFT keypoints, we could also require that the scales and orientations of the keypoints are also consistent with the homography $\mathbf{H}$. (Details omitted.)

   If the consensus set is sufficiently large, then we could refit the model using all points in the consensus set. If the fit is the best of the models tested so far, then we can save the model.

3. Repeat step 2 a certain number of times or until we find a homography whose consensus set exceeds some pre-determined threshold).

Case 3 revisited: image stitching

I finished the lecture by describing how to remap image intensities using a homography. This is necessary for stitching two images together to make a panorama, for example.

Suppose we have two images $I(x, y)$ and $I'(x', y')$ and we have a homography $\mathbf{H}$ that maps (x, y) to $(x',' y')$:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We would like to combine these two images together to get a larger image.

Suppose we start with image $I(x, y)$ and we expand this image by adding black (or unassigned) pixels outside the current domain of the image. For each of these unassigned pixels, we then apply the homography to find a corresponding position (x', y') in the other image. If this (x', y') falls in the domain of the other image, then we can take the value $I'(round(x'), round(y')$ of the closest pixel or interpolate the values from the nearest neighbors (since the mapped location will unlikely land exactly on a pixel). If the (x', y') falls outside the domain of the other image, then we would have to leave the pixel unassigned (or some default value like white or black).

