At the end of the lecture 19 notes, I introduced homographies and gave two examples. Here I continue with another example.

More homographies (panoramas and image stitching)

Homographies can arise for *general scenes* as well (non-planar), 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 often done is modern digital cameras when you stitch images together to make a panorama.

To see that two images taken under these conditions 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

$$\begin{bmatrix} wx_1 \\ wy_2 \\ w \end{bmatrix} = \mathbf{K}\mathbf{R}_1[\mathbf{I}|-\mathbf{c}] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

and

$$\begin{bmatrix} w'x_2 \\ w'y_2 \\ w' \end{bmatrix} = \mathbf{K}\mathbf{R}_2[\mathbf{I}|-\mathbf{c}] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

for the two orientations of the camera. Then,

$$(\mathbf{K}\mathbf{R}_1)^{-1} \begin{bmatrix} wx_1 \\ wy_1 \\ w \end{bmatrix} = (\mathbf{K}\mathbf{R}_2)^{-1} \begin{bmatrix} w'x_2 \\ w'y_2 \\ w' \end{bmatrix}$$

and so

$$\begin{bmatrix} w'x_2 \\ w'y_2 \\ w' \end{bmatrix} = \mathbf{K}\mathbf{R}_2(\mathbf{K}\mathbf{R}_1)^{-1} \begin{bmatrix} wx_1 \\ wy_1 \\ w \end{bmatrix}$$

Thus, we have a homography $\mathbf{K}\mathbf{R}_2(\mathbf{K}\mathbf{R}_1)^{-1}$ that takes pixels (x_1, y_1) in camera orientation 1 to pixels (x_2, y_2) 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.

Image rectification

Next week 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. Next week we'll 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 orientation 1, e.g. if $\mathbf{T}$ were $(1, 0, 0)^T$ then $\mathbf{T}$ would be in the camera's X axis direction.

Let $\hat{\mathbf{z}} = (0, 0, 1)$ be camera 1's optical axis direction. Define the rotation matrix, which rotates $\mathbf{T}$ to become the new X axis direction for camera 1,

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

where *unit* is the operator that divides a vector by its length. By inspection, the rows of this matrix are orthonormal and indeed

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

We would next like 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 orientation to camera 1's orientation, we apply the rotation $\mathbf{R}_{rect}\mathbf{R}_1\ \mathbf{R}_2^{-1}$. We now have rotations that bring the cameras into a nice orientation where their XYZ axes are parallel and the X axis is in the direction from camera 1 to camera 2. The next step is to re-map the pixels so they also have a nice correspondence. This is the *rectification* step.

For any pixel (x_1, y_1) in the original camera 1 image, we undo the camera projection matrix to write the point in mm instead of in pixels), then rotate, and then remap to pixels.

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

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' x axis direction.

We will return to this configuration next week, when we discuss the stereo problem. In the remainder of this lecture, we will discuss one final problem that involves homographies, namely given two images with a set of keypoints, how do you find a homography that maps from one image to the other?

Solving for the homography between two images

Suppose we have N sets of corresponding point pairs (x_i, y_i) and $(\tilde{x}_i, \tilde{y}_i)$ between two images which are *approximately* related by an *unknown* homography $\mathbf{H}$, namely

$$\begin{bmatrix} w\tilde{x}_i \\ w\tilde{y}_i \\ w \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

The points might be image features such as Harris corners. Note that we don't expect to localize these points exactly - there may be errors.

Using all N points

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

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -\tilde{x}_1 x_1 & -\tilde{x}_1 y_1 & -\tilde{x}_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -\tilde{y}_1 x_1 & -\tilde{y}_1 y_1 & -\tilde{y}_1 \\ : & : & : & : & : & : & : & : & : \\ x_N & y_N & 1 & 0 & 0 & 0 & -\tilde{x}_N x_N & -\tilde{x}_N y_N & -\tilde{x}_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -\tilde{y}_N x_N & -\tilde{y}_N y_N & -\tilde{y}_N \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ : \\ 0 \\ 0 \end{bmatrix}$$

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.

[ASIDE: The relationship between this problem and the one we looked at last class is closer than you might think. When we had a calibration grid, we knew a set of points (X_i, Y_i, Z_i). In the case that all points lie on a plane, we can think of the plane as $Z_i = 0$ 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 exactly the problem above.]

Using 4 points only

If we have four corresponding points $(x_i, y_i, \tilde{x}_i, \tilde{y}_i), i = 1, \ldots, 4$ then we get a system of 8 homogenous equations with 9 unknowns. A homography $\mathbf{H}$ that puts the points into *exact* correspondence can be obtained by finding the null space of this system of equation. That is, we have 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 $\mathbf{A}$.

Using RANSAC

In the context of an autonomous vision system such as a robot, we would like to automate this computation, including the problem of finding matching points. This may be difficult to do correctly. Why? If we have two cameras and the scene contains a plane (like last lecture), there may be other surfaces in the scene that do not belong to this plane and that would not be explained by the homography – these other points need somehow to be automatically avoided. Even if you can avoid such "outliers", it may still be difficult to find correct matches for points on the plane since the plane is viewed by two different cameras, e.g. a bright red flower might be a good feature to detect match from one image to the other, but there might be many bright red flowers.

This should remind you of a problem we saw in lecture 15 where we tried to fit points to lines. Indeed similar approaches are used for homographies. Here is the typical RANSAC approach:

1. Find many feature points (e.g. Harris corners, or SIFT features) in each of the two images: $\{(x_i, y_i)\}$ and $\{(x'_j, y'_j)\}$.

2. For each feature point (x_i, y_i) in one image, find a candidate corresponding feature point in the second image (x_i', y_i') whose intensity neighborhood is similar e.g. use SIFT descriptors. This give a set of 4-tuples (x_i, y_i, x_i', y_i'). You might want to allow each (x_i, y_i) to have several candidate matches (x_i', y_i') and vice-versa.

3. Randomly choose four 4-tuples and fit an exact homography $\mathbf{H}_i$ that maps the four $\{(x_i, y_i)\}$ exactly to their corresponding $\{(x_j', y_j')\}$. 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 $(x, y, \ tildex, \ tildey$ be a new 4-tuple and we test whether it fits a homography $\mathbf{H}_i$ by computing

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

   and defining the distance/fit to be

$$dist_{\mathbf{H}_i}(x, y, \tilde{x}, \tilde{y}) \ = \ \| \ (\tilde{x}, \tilde{y}) - (x', y') \ \|_2 \ .$$

4. After repeating step 3 a certain number of times (or until you find a homography whose consensus set exceeds some pre-determined threshold), choose the homography which yielded the largest consensus set and use that consensus set to re-estimate a homography $\mathbf{H}$ using least squares. The reason you do this final step is that the homographies $\mathbf{H}_i$ were chosen to exactly fit four pairs of points, and if there were any noise in those points (likely) then the fit would be biased by the particular noise of those points, which is undesirable.

Image remapping using homographies

I finished the lecture by describing how to remap image intensities using homographies. Please see the lecture slides. You will need to use this method for Assignment 4. (I am not specifying every little detail here because I would like you to think it some of it through for yourself. The slides give you most of what you need.)