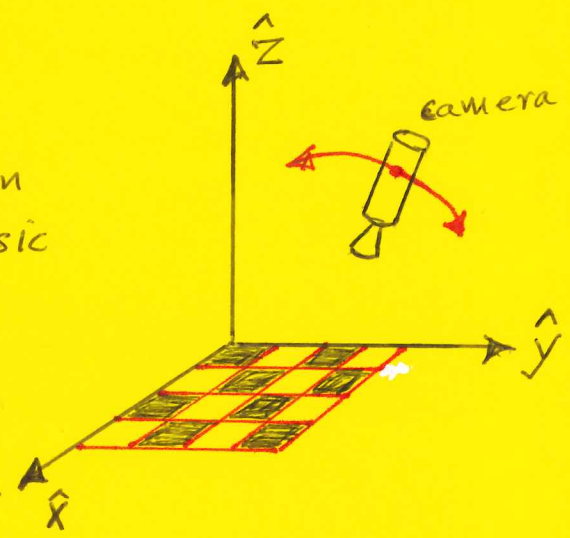
ECE 661
Purdue Univ.
Avi Kak
Lecture 19

Camera Calibration -Zhang's Algorithm

- The camera calibration algorithm developed originally by Zhengyou Zhang (of Microsoft Research) on the basis of the ideas discussed in the previous lecture is the world's most popular such algorithm today.

- The algorithm yields both the intrinsic parameters of a camera, as represented by the elements of its 3x3 matrix $\mathbf{K}$, and its extrinsic parameters R and $\vec{t}$ with respect to a chosen frame of reference.

- The algorithm assumes you have a calibration pattern in the $Z=0$ plane of the world frame and that you are taking images of this pattern from different viewpoints. Obviously, the extrinsic parameters you calculate would be for one such viewpoint. (This assumption generalizes trivially to the case when the camera is mounted in a fixed position and the pattern moved about.) The calibration pattern is needed in order to estimate the homography from the $Z=0$ plane to the image plane of the camera.



- Denoting the pixel coordinates by the homogeneous vectors $\vec{x} = (x, y, w)^T$ and the world points by the homogeneous vectors $\vec{X} = (X, Y, Z, W)^T$, we have $\boxed{\vec{x} = K[R|\vec{t}]\begin{pmatrix} X \\ Y \\ 0 \\ W \end{pmatrix} = H\vec{X}_M}$ where $\vec{X}_M = (X, Y, W)^T$. The subscript 'M' refers to the model, meaning the calibration pattern. We will write $H = [\vec{h}_1 \; \vec{h}_2 \; \vec{h}_3]$. For any R and $\vec{t}$, we can find the column vectors $\vec{h}_1$, $\vec{h}_2$, and $\vec{h}_3$ by estimating the homography H.

- Zhang's algorithm is based on the fact that, as you saw in Lecture 18, the camera image of the Absolute Conic Ω_∞ is independent of R and $\vec{t}$, and that it is given by $\omega = K^{-T}K^{-1}$. Furthermore, we know from Lecture 18 that any plane in the world frame (such as the $Z=0$ plane) samples Ω_∞ at exactly two points. The images of these two points fall on the conic ω in the camera image plane. Each of these two points must obey the conic constraint $\vec{x}^T\omega\vec{x} = 0$. As you saw in Lecture 18, when we plug the coordinates of the two image points in the conic constraint equation, we get $\boxed{\vec{h}_1^T\omega\vec{h}_1 = \vec{h}_2^T\omega\vec{h}_2}$ and $\boxed{\vec{h}_1^T\omega\vec{h}_2 = 0}$. Given $\vec{h}_1$ and $\vec{h}_2$ for a number of different positions of the camera, our goal is to estimate ω and from there to estimate the elements of K.
[Here is another way to arrive at the two red-boxed equations: From the red-boxed equation in the fourth bullet above, we have $H = K[\vec{r}_1 \; \vec{r}_2 \; \vec{t}]$ where we have used $R = [\vec{r}_1 \; \vec{r}_2 \; \vec{r}_3]$. We get $K^{-1}[\vec{h}_1 \; \vec{h}_2 \; \vec{h}_3] = [\vec{r}_1 \; \vec{r}_2 \; \vec{t}]$. We know that R is a rotation matrix, which means $\|\vec{r}_1\| = \|\vec{r}_2\|$ and $\vec{r}_1^T\vec{r}_2 = 0$. Substituting $\vec{r}_1 = K^{-1}\vec{h}_1$

and $\vec{r}_2 = K^{-1}\vec{h}_2$ in the constraints on the columns of R, we get the same two equations as shown in the two red boxes.]

- The essence of Zhang's algorithm consists of: ① How the different pairs of equations $\vec{h}_1^T \omega \vec{h}_1 = \vec{h}_2^T \omega \vec{h}_2$ and $\vec{h}_1^T \omega \vec{h}_2 = 0$, one pair for each position of the camera vis-a-vis the Z=0 plane, are stacked together for solving for the elements of the image conic ω; ② How the elements of K are extracted from the elements of ω; ③ How the initial estimates of R and $\vec{t}$ are constructed for every position of the camera vis-a-vis the Z=0 plane; and ④ How the calibration parameters are refined and the rotation matrices conditioned.

- Let's now focus on the unknowns in $\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix}$. Since an image conic in its homogeneous representation is always a 3×3 symmetric matrix, we have only six unknowns in ω. We will express these through a vector $\vec{b} = \begin{pmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{22} \\ \omega_{13} \\ \omega_{23} \\ \omega_{33} \end{pmatrix}$

- Let's now express the equations $\vec{h}_1^T \omega \vec{h}_1 - \vec{h}_2^T \omega \vec{h}_2 = 0$ and $\vec{h}_1^T \omega \vec{h}_2 = 0$ in terms of the vector $\vec{b}$ of unknowns. To see how that can be done, we note $\vec{h}_1^T \omega \vec{h}_2 = 0 \Rightarrow (h_{11}\ h_{12}\ h_{13}) \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix} = 0 \Rightarrow \boxed{\vec{v}_{12}^T \vec{b} = 0}$ where the vector of knowns $\vec{v}_{ij}$ is given by the form on the right with i=1 and j=2: $\vec{v}_{ij} = \begin{pmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{pmatrix}$

- In the same manner, the equation $\vec{h}_1^T \omega \vec{h}_1 - \vec{h}_2^T \omega \vec{h}_2 = 0$ can be expressed as $\boxed{(\vec{v}_{11} - \vec{v}_{22})^T \vec{b} = 0}$.

- Therefore, the two equations $\vec{h}_1^T \omega \vec{h}_2 = \vec{h}_2^T \omega \vec{h}_2$ and $\vec{h}_1^T \omega \vec{h}_2 = 0$ together can be expressed as $\boxed{V\vec{b} = \vec{0}}$ where the matrix V is given by $V = \begin{bmatrix} \vec{v}_{12}^T \\ (\vec{v}_{11} - \vec{v}_{22})^T \end{bmatrix}$ ← 2 rows, 6 columns. a 2×6 matrix of known values

- Each position of the camera vis-a-vis the Z=0 plane gives us 2 equations represented by $V\vec{b} = \vec{0}$. Since we have 6 unknowns, we need at least 3 camera positions. Using V_i to represent the 2×6 matrix V for the i^{th} camera position, we can stack up all the resulting equations as: $\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} \vec{b} = \vec{0} \Rightarrow \boxed{V\vec{b} = \vec{0}}$ ← 2n×6, where n is the number of camera positions used. These can be solved for $\vec{b}$ by using linear least-squares method of Lecture 11.

- Once we have calculated $\vec{b}$, we know the image conic ω. Our next goal is to find the elements of the K matrix from the ω matrix. Recall $\omega = K^{-T}K^{-1}$.

Calculating the Intrinsic Parameters of the Camera

- Now that we have the image ω of the Absolute Conic, it is time to extract the intrinsic parameters of the camera from ω. Recall $K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$

- From $\omega = K^{-T}K^{-1}$, we have:

$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix} \triangleq \begin{bmatrix} \frac{1}{\alpha_x^2} & \frac{-s}{\alpha_x^2 \alpha_y} & \frac{y_0 s - x_0 \alpha_y}{\alpha_x^2 \alpha_y} \\ \frac{-s}{\alpha_x^2 \alpha_y} & \frac{s^2}{\alpha_x^2 \alpha_y^2} + \frac{1}{\alpha_y^2} & \frac{-s(y_0 s - x_0 \alpha_y)}{\alpha_x^2 \alpha_y^2} - \frac{y_0}{\alpha_y^2} \\ \frac{y_0 s - x_0 \alpha_y}{\alpha_x^2 \alpha_y} & \frac{-s(y_0 s - x_0 \alpha_y)}{\alpha_x^2 \alpha_y^2} - \frac{y_0}{\alpha_y^2} & \frac{(y_0 s - x_0 \alpha_y)^2}{\alpha_x^2 \alpha_y^2} + \frac{y_0^2}{\alpha_y^2} + 1 \end{bmatrix}$$

- There is an important reason for why the equality in the previous bullet was depicted with the symbol '$\triangleq$': Whereas the entity on the left of the symbol is homogeneous, the entity on the right is inhomogeneous. Therefore, multiplying all of the elements of the ω matrix by the same non-zero scalar does not alter the image conic that ω represents. We cannot say the same for the ~~entire~~ 3×3 matrix on the right of '$\triangleq$'. Therefore, it is NOT possible to solve for the intrinsic parameters by setting element-by-element equalities between the matrices on the two sides of '$\triangleq$'.

- Our goal is to scale the elements of the ω matrix such that the final solution the K matrix looks like: $K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$. Zhang has shown how this scaling of ω is taken care of implicitly by introducing a new variable λ as shown by the following solution for the intrinsic parameters:

$$x_0 = \frac{\omega_{12}\omega_{13} - \omega_{11}\omega_{23}}{\omega_{11}\omega_{22} - \omega_{12}^2}$$

$$\lambda = \omega_{33} - \frac{\omega_{13}^2 + x_0(\omega_{12}\omega_{13} - \omega_{11}\omega_{23})}{\omega_{11}}$$

$$\alpha_x = \sqrt{\frac{\lambda}{\omega_{11}}}$$

$$\alpha_y = \sqrt{\frac{\lambda\,\omega_{11}}{\omega_{11}\omega_{22} - \omega_{12}^2}}$$

$$s = -\frac{\omega_{12}\alpha_x^2\alpha_y}{\lambda}$$

$$y_0 = \frac{s x_0}{\alpha_y} - \frac{\omega_{13}\alpha_x^2}{\lambda}$$

- It is interesting to note that much of the "complexity" of the expressions in the expanded form of the $K^{-T}K^{-1}$ matrix at the bottom of the previous page is owing to the skew parameter s. When s can be assumed to be zero for a camera, we have $K^{-1} = \begin{bmatrix} 1/\alpha_x & 0 & -x_0/\alpha_x \\ 0 & 1/\alpha_y & -y_0/\alpha_y \\ 0 & 0 & 1 \end{bmatrix}$. Multiplying this matrix on the left by its transpose yields a simpler form for $K^{-T}K^{-1}$.

Calculating the Extrinsic Parameters of the Camera

- We now calculate the extrinsic parameters R and $\vec{t}$ of the camera for one of its positions vis-a-vis the $Z=0$ plane that holds the calibration pattern. (If needed, we can do this for every position of the camera.)

- For the extrinsic parameters, we go back to the relationship $\boxed{K^{-1}[\vec{h}_1\,\vec{h}_2\,\vec{h}_3] = [\vec{r}_1\,\vec{r}_2\,\vec{t}]}$ that was shown at the bottom of page 19-1. The vectors $\vec{h}_1, \vec{h}_2, \vec{h}_3$ are the columns of the homography H that was estimated for the position of the camera for which you seek $R = [\vec{r}_1\ \vec{r}_2\ \vec{r}_3]$ and $\vec{t}$.

- Since the equation in the red box above involves a homogeneous entity, which is "immune" to multiplication by a non-zero scalar, and an inhomogeneous entity, we must be careful about how we interpret the equality. We first set

$$\vec{r}_1 = K^{-1}\vec{h}_1$$
$$\vec{r}_2 = K^{-1}\vec{h}_2$$
$$\vec{r}_3 = \vec{r}_1 \times \vec{r}_2$$
$$\vec{t} = K^{-1}\vec{h}_3$$

- At this point we have no reason to expect that the vectors calculated as shown at right have the correct scaling. Fortunately, we can discover the scale factor from the orthonormality property of the R matrix. This property says that all columns (as also all rows) must be of unit magnitude. Therefore, the scale factor must be $\xi_1 = \frac{1}{\|K^{-1}\vec{h}_1\|}$ (with the expectation that $\frac{1}{\|K^{-1}\vec{h}_2\|}$ would yield the same value.) So, the actual formulas we use for R and $\vec{t}$ become: $\vec{r}_1 = \xi_1 K^{-1}\vec{h}_1$, $\vec{r}_2 = \xi_1 K^{-1}\vec{h}_2$, $\vec{r}_3 = \vec{r}_1 \times \vec{r}_2$, $\vec{t} = \xi_1 K^{-1}\vec{h}_3$

Refining the Calibration Parameters

- The camera calibration algorithm develope up to this point started with a linear least-squares solution to the homogeneous equations $V\vec{b}=0$ as shown on page 19-2. The algebraic minimization in the linear least-squares solution does NOT give us an intuitive sense of how accurate our calculated calibration parameters are.

- To develop a more intuitively accessible measure of the accuracy of the calculated camera parameters, we proceed as follows: For each position of the camera used for calibration, we form the projection matrix P using the calculated calibration parameters. Whereas K will be the same for all positions of the camera, the extrinsic parameters R and $\vec{t}$ will be specific to each. We use these P matrices to project a set of salient points from the calibration pattern in the Z=0 plane into the image planes for each of the camera positions. The Euclidean distance between the projected pixels and where they actually occur in the images gives us a meaningful metric for assessing the quality of the calculated calibration parameters. In what follows, let's develop an analytic expression for this metric.

- NOTATION:
  - $\vec{x}_{M,j}$: The j^{th} salient point on the calibration pattern
  - $\vec{x}_{i,j}$: The actual image point for $\vec{x}_{M,j}$ in the i^{th} position of the camera
  - $\hat{\vec{x}}_{i,j}$: The projected image point for $\vec{x}_{M,j}$ using P for i^{th} camera position
  - R_i : The rotation matrix for the i^{th} position of the camera.
  - $\vec{t}_i$: The translational vector for the i^{th} position of the camera.
  - K : The camera calibration matrix for the intrinsic parameters

  Important: The vectors $\vec{x}_{M,j}$, $\vec{x}_{ij}$, $\hat{\vec{x}}_{ij}$ represent the actual physical coordinates (as opposed to homogeneous coordinates).

- If we could assume that the calculated camera calibration parameters have zero error, we'll have $\vec{x}_{ij} = \hat{\vec{x}}_{ij}$. In general, the Euclidean distance $\| \vec{x}_{ij} - \hat{\vec{x}}_{ij} \|$ tells us something about how far the calibration is with respect to the j^{th} salient point in the i^{th} position of the camera. We can visualize this Euclidean distance by looking at images that show both the projected and the actually imaged calibration pattern salient points.

- Aggregating the error distances for all salient points and for all the camera positions, we have $d^2_{geom} = \sum_i \sum_j \| \vec{x}_{ij} - \hat{\vec{x}}_{ij} \|^2 = \sum_i \sum_j \| \vec{x}_{ij} - K[R_i | t_i] \vec{x}_{M,j} \|^2$ ($K[\vec{r}_{i,1}\ \vec{r}_{i,2}\ \vec{t}_i]$)
  Note that we have used the squares of Euclidean distances in the overall quality metric d^2_{geom}. Using the squares allows us to write $d^2_{geom} = \| \vec{X} - \vec{f}(\vec{p}) \|^2$. That's because the square of a Euclidean distance is equal to the sum of the squares of the x-coord difference and the y-coord difference. This allows us to treat all of the x-coordinates and all of y-coordinates as the individual elements of a single large vector $\vec{X}$. And the same goes for the coordinates of $\hat{\vec{x}}_{ij}$.

- The quality measure d_{geom} is our overall geometric distance between the image pixels for the salient points on the calibration pattern and where they are projected to be on the basis of the calculated camera parameters.

- Using the form $\|\vec{X} - \vec{f}(\vec{p})\|^2$ for d^2_{geom} allows us to use the methods of nonlinear least-squares minimization as described in Lecture 12 for refining the values of the camera parameters. Note $\vec{p} = (K, R_i, \vec{t}_i \,|\, i=1,2,\ldots)^T$, where we obviously mean to write out all of the variables in K, R_i, and $\vec{t}$ as a sequence.

- It is interesting to note that the solution you get by minimizing d^2_{geom} can be considered to be the Maximum-Likelihood solution (ML). That is because the discrepancy between the actually measured $\vec{x}_{ij}$ values and the projected $\hat{\vec{x}}_{ij}$ values can be modeled as a bivariate Gaussian distribution as given by

$$prob(\vec{x}_{ij} \,|\, \hat{\vec{x}}_{ij}) \equiv prob(\vec{x}_{ij} \,|\, \vec{p}) = \frac{1}{2\pi\sigma^2} e^{-\frac{\|\vec{x}_{ij} - \hat{\vec{x}}_{ij}\|^2}{2\sigma^2}} \quad \text{for some value of } \sigma.$$

If we assume that such discrepancies are uncorrelated for different i and j,

$$prob(\{\vec{x}_{ij} \,|\, i,j=1,2,\ldots\} \,|\, \vec{p}) = \prod_{i,j} \frac{1}{2\pi\sigma^2} e^{-\frac{\|\vec{x}_{ij} - \hat{\vec{x}}_{ij}\|^2}{2\sigma^2}} = \left(\prod_{i,j} \frac{1}{2\pi\sigma^2}\right) e^{-\sum_{ij} \frac{\|\vec{x}_{ij} - \hat{\vec{x}}_{ij}\|^2}{2\sigma^2}} = \boxed{A e^{-\frac{d^2_{geom}}{2\sigma^2}}}$$

for some value of the constant A. If we take the natural log of both sides, we get $\boxed{\ln prob(\vec{X} \,|\, \vec{p}) = \ln A - \frac{d^2_{geom}}{2\sigma^2}}$. An estimate for the parameters $\vec{p}$ is considered to be ML if it maximizes the probability of the observed data as being $\vec{X}$. Obviously, the smaller the value of d^2_{geom}, the larger the log-likelihood on the left.

- There is one more issue of CRITICAL IMPORTANCE that you must understand before using any of the Lecture 12 methods for the minimization of $\|\vec{X} - \vec{f}(\vec{p})\|^2$. This issue concerns the representation of each R_i in the parameter vector $\vec{p}$. You see, each $R_i = \begin{bmatrix} r_{i,11} & r_{i,12} & r_{i,13} \\ r_{i,21} & r_{i,22} & r_{i,23} \\ r_{i,31} & r_{i,32} & r_{i,33} \end{bmatrix}$ has 9 elements but only 3 DoF. In any optimization algorithm, the number of variables used to represent an entity must equal (strictly) the DoF of the entity. If you minimize $\|\vec{X} - \vec{f}(\vec{p})\|^2$ willy-nilly while allocating 9 elements of $\vec{p}$ to each R_i, you are likely to get absurd results for R_i.

- What we need is a 3-parameter representation of a rotation matrix R. A commonly used such representation is known as the Rodrigues Representation in which a rotation in 3D is expressed as a vector $\vec{w} = \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$ such that the direction of the axis of the rotation is encoded in the unit vector $\frac{\vec{w}}{\|\vec{w}\|}$ and the angle φ of clockwise rotation around this axis by the magnitude $\varphi = \|\vec{w}\|$.

- What we need next is a way to go back and forth between R and $\vec{w}$ for representing the same rotation. In order to go from a given $\vec{w}$ to R, you need to first represent $\vec{w}$ by the 3x3 matrix $[\vec{w}]_X = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$. (One can show easily that for any vector $\vec{z}$, the cross-product $\vec{w} \times \vec{z}$ is the same as the matrix-vector product $[\vec{w}]_X \vec{z}$. Hence the notation $[\vec{w}]_X$.)

- To construct the R matrix for a given vector $\vec{w}$, we use the equation

$$R = e^{[\vec{w}]_X} = I_{3\times 3} + \frac{\sin\varphi}{\varphi}[\vec{w}]_X + \frac{1-\cos\varphi}{\varphi^2}[\vec{w}]_X^2 \quad \text{where } \varphi = \|\vec{w}\|.$$

And to construct the vector $\vec{w}$ for a given R, we first write for the angle φ:

$$\varphi = \cos^{-1}\frac{\text{trace}(R)-1}{2}.$$

Subsequently,

$$\vec{w} = \frac{\varphi}{2\sin\varphi}\begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{pmatrix} \quad \text{with } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

Converting $\vec{w}$ to R guarantees orthonormality of R.

Incorporating Radial Distortion

- The camera calibration procedure discussed up to this point is based on the pinhole model of camera imaging as presented in Lecture 16. The pinhole model is a linear model — linear in homogeneous coordinates. The pinhole model is linear in another sense also: straight lines in the world 3D are imaged as straight lines by the camera. The pinhole model is a good approximation for most cameras we use on an everyday basis. More specifically, the linear model works well for long focal length cameras.

- The pinhole model breaks down for short focal-length cameras. If a camera images a perfectly rectangular object like  as  , it is exhibiting what is known as **radial distortion.**

- To construct a model for a camera that exhibits radial distortion, you start with a pinhole model for the camera — under the assumption that the pinhole principle would dominate in determining where a pixel for a given object point would show up. **Subsequently, you adjust radially the pixel coordinates predicted by the pinhole model.** Typically, this radial adjustment involves a couple of new parameters. Determining these parameters becomes a part of the overall calibration procedure.

- To calibrate a camera with radial distortion, you carry out all of the steps on pages 19-1 through 19-3 to find tentative values for the intrinsic parameters in K and the extrinsic parameters $(R_i, \vec{t}_i)$ for each position of the camera used for calibration. Now let $(\hat{x}, \hat{y})$ be the predicted position of a pixel using the pinhole model and the tentative set of calibration parameters. And let $(\hat{x}_{rad}, \hat{y}_{rad})$ be the pixel coordinates that would be predicted if you included the radial distortion. You say

$$\begin{aligned} \hat{x}_{rad} &= \hat{x} + (\hat{x}-x_0)[k_1 r^2 + k_2 r^4] \\ \hat{y}_{rad} &= \hat{y} + (\hat{y}-y_0)[k_1 r^2 + k_2 r^4] \end{aligned}$$

where (x_0, y_0) are the currently available for the principal point on the image plane of the camera and where $r^2 = (\hat{x}-x_0)^2 + (\hat{y}-y_0)^2$. As you can see, the radial displacement of a pinhole-predicted pixel depends on how far the pixel is radially from the principal point of the camera.

- Our next goal is to invoke the nonlinear least-squares part of the

calibration procedure (as presented on pages 19-4 and 19-5) to not only refine the previously calculated values for K and for $\{R_i, \vec{t}_i \mid i=1,2,\ldots\}$, but to also estimate values for the new parameters k_1 and k_2 that characterize the radial distortion. We initialize k_1 and k_2 to 0.

- To make the estimation of k_1 and k_2 a part of the minimization of $d^2_{geom} = \| \vec{X} - \vec{f}(\vec{p}) \|^2$ we note that for every actually measured pixel coordinate pair in $\vec{X}$, $\vec{f}(\vec{p})$ is supposed to provide a prediction of those coordinates using the camera model. So all we have to do is to chain pinhole modeling with the radial-distortion modeling as described at the bottom of the previous page. In other words, the prediction function $\vec{f}(\vec{p})$ will first calculate the pixel coordinates for each salient point on the calibration pattern using just the pinhole model, and then invoke the radial distortion model to find the final pixel coordinates. This would automatically make estimating k_1 and k_2 a part of the minimization procedure.

Conditioning the Rotation Matrix

- It is likely that the procedure described on page 19-3 for the calculation of the extrinsic parameters will result in rotation matrices R_i that are NOT orthonormal. (Recall, we use the same ξ for the normalization of both $K^{-1}\vec{h}_1$ and $K^{-1}\vec{h}_2$. Additionally, as calculated, $\vec{h}_1$ and $\vec{h}_2$ may not be strictly orthogonal.) Therefore, before converting each R_i into a Rodriguz vector $\vec{w}_i$, we must condition the R_i to make it orthonormal.
- To show how a rotation matrix can be conditioned, let Q represent the computed rotation matrix. Our goal is to find the best orthonormal approximation R to Q.
- For a given Q, we will find R by solving the following problem:

$$\min_R \| R - Q \|_F^2 \quad \text{subject to} \quad R^T R = I$$

where $\|\circ\|_F$ is the Frobenius norm given by $\|R-Q\|_F^2 = \text{trace}\left((R-Q)^T(R-Q)\right)$. (In general, the Frobenius norm of an $m \times n$ matrix A is: $\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2 = \text{tr}(AA^T)$)

- To solve the problem stated above, we note $\text{tr}\left((R-Q)^T(R-Q)\right) = \text{tr}\left(R^TR - R^TQ - Q^TR + Q^TQ\right) = \text{tr}\left(I - R^TQ - Q^TR + Q^TQ\right) = 3 - 2\,\text{tr}(R^TQ) + \text{tr}(Q^TQ)$. Therefore, our goal of finding R that minimizes $\text{tr}\left((R-Q)^T(R-Q)\right)$ can be restated as finding that R which maximizes $\text{tr}(R^TQ)$.
- In what follows, we'll use the following two properties of the trace of a matrix: (1) $\text{tr}(AB) = \text{tr}(BA)$; and (2) If D is a diagonal matrix $D = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$, $\text{tr}(AD) = \sum_{i=1}^{3} \sigma_i a_{ii}$ if $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$.
- Let Q's SVD be $Q = UDV^T$ with $D = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$. Since Q is square and real,

both U and V are orthonormal and the column vectors of V are the eigenvectors of Q^TQ. When a matrix is square and real, its SVD boils down to a rotation by V^T, followed by a scaling of the axes by D, followed by another rotation by U.

- Our goal is to find an orthonormal R that maximizes $tr(R^TQ)$. We write $tr(R^TQ) = tr(R^TUDV^T) = tr(V^TR^TUD) = tr(ZD)$ where $Z = V^TR^TU$.
  As it turns out, Z is orthonormal since $Z^TZ = (V^TR^TU)^TV^TR^TU = U^TRVV^TR^TU = I$
  Let $Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix}$. Z being orthonormal implies that the norm of each of its column vectors cannot exceed 1. That means no element Z can exceed 1. So $z_{ii} \leq 1$ for $i=1,2,3$.

- We know from the second to the last bullet on the previous page that $tr(ZD) = \sum_{i=1}^{3} \sigma_i z_{ii}$. In the expression on the right, σ_i's are fixed for a given Q. However, the values of z_{ii} depend on our choice of R. We want to choose that R which gives z_{ii}'s their largest possible value of 1. When you combine that with the orthonormality of Z, you conclude that the best choice for R must make Z ~~a diagonal~~ an identity matrix. So $Z = I$.

- Our choice of R must therefore satisfy $V^TR^TU = I$. Setting $\boxed{R = UV^T}$ satisfies this condition since $V^T(UV^T)^TU = V^TVU^TU = I \cdot I = I$.

- CONCLUSION: To condition a rotation matrix, you calculate its SVD, and you re-set all the singular values to 1.

Degenerate Configurations of the Camera vis-a-vis the Calibration Pattern

- The issue here is: As we change the extrinsic relationship between the camera and the calibration pattern, are there configurations that would NOT yield independent information regarding the camera parameters?

- It is easy to show that if you merely rotate the camera around the Z-axis, in the figure on page 19-1, you will obtain linearly dependent 2-point samples on the image ω of the Absolute Conic. ~~Let~~ Let $R^{(1)} = [\vec{r}_1^{(1)}\ \vec{r}_2^{(1)}\ \vec{r}_3^{(1)}]$ be the rotation part of the homography $H^{(1)} = [\vec{h}_1^{(1)}\ \vec{h}_2^{(1)}\ \vec{h}_3^{(1)}]$ for a given position of the camera vis-a-vis the $Z=0$ plane. Similarly, let $R^{(2)} = [\vec{r}_1^{(2)}\ \vec{r}_2^{(2)}\ \vec{r}_3^{(2)}]$ be the rotation part of the homography $H^{(2)} = [\vec{h}_1^{(2)}\ \vec{h}_2^{(2)}\ \vec{h}_3^{(2)}]$ when the camera is rotated through angle θ around the Z axis. We have $R^{(2)} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} R^{(1)}$. It follows from the arguments at the bottom of page 19-3, that

$$\vec{h}_1^{(2)} = \frac{K}{\xi_1}\vec{r}_1^{(2)} = \frac{K}{\xi_1}\left(\vec{r}_1^{(1)}\cos\theta - \vec{r}_2^{(1)}\sin\theta\right)$$
$$\vec{h}_2^{(2)} = \frac{K}{\lambda}\vec{r}_2^{(2)} = \frac{K}{\xi_1}\left(\vec{r}_1^{(1)}\cos\theta + \vec{r}_2^{(1)}\sin\theta\right)$$

This implies →
$$\vec{h}_1^{(2)T}\omega\vec{h}_2^{(2)} = \frac{1}{\xi_1}\left[(\cos^2\theta - \sin^2\theta)\,\vec{h}_1^{(1)T}\omega\vec{h}_2^{(1)} - \cos\theta\sin\theta\left(\vec{h}_1^{(1)T}\omega\vec{h}_1^{(1)} - \vec{h}_2^{(1)T}\omega\vec{h}_2^{(1)}\right)\right]$$

The conic eqn in the 2nd camera position depends linearly on the eqns for the 1st position

- In general, just translating the camera also gives us degenerate configurations