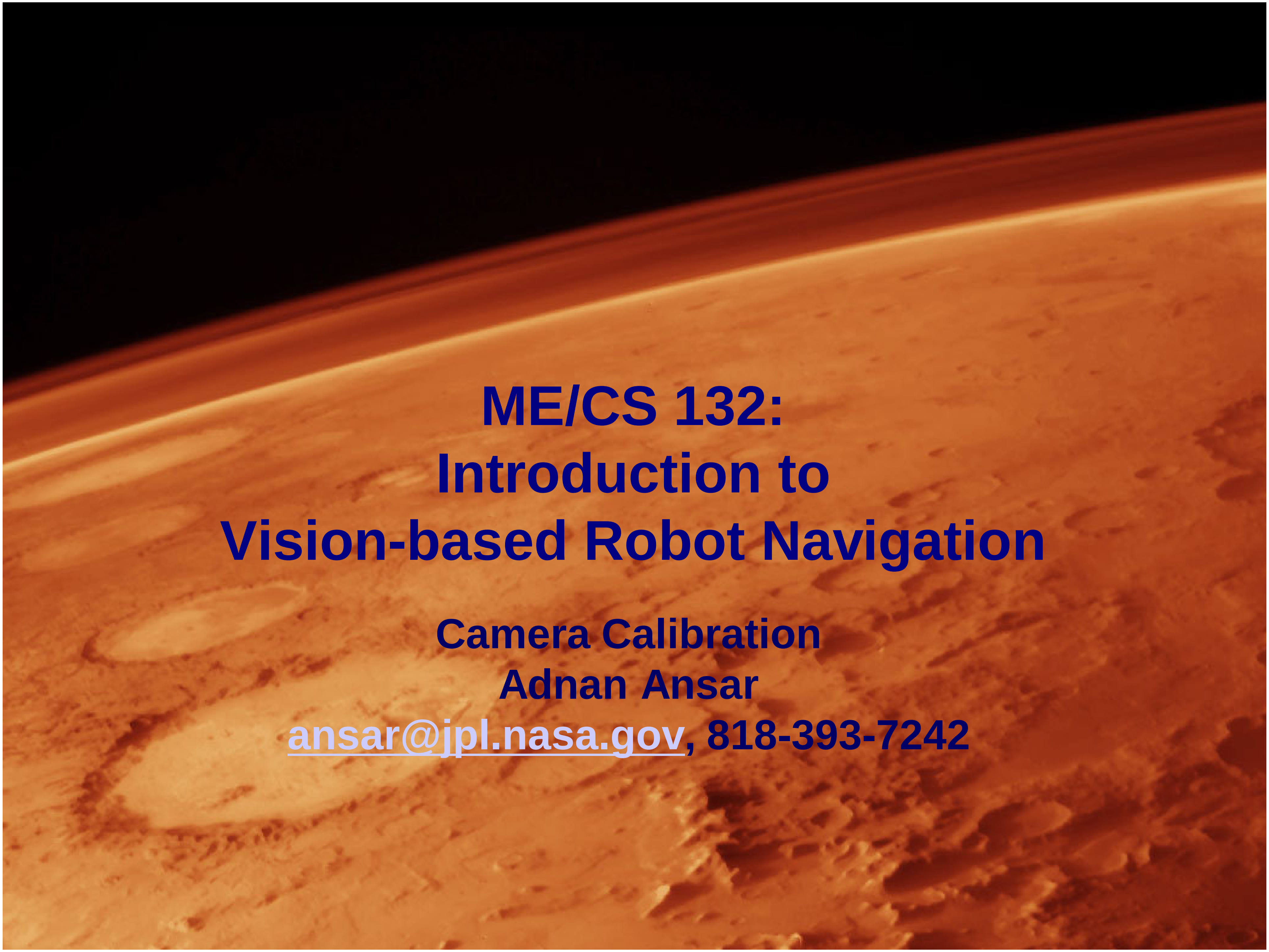


ME/CS 132: Introduction to Vision-based Robot Navigation

Camera Calibration

Adnan Ansar
ansar@jpl.nasa.gov, 818-393-7242

Perspective Projection Revisited

- Assume pinhole model of a camera

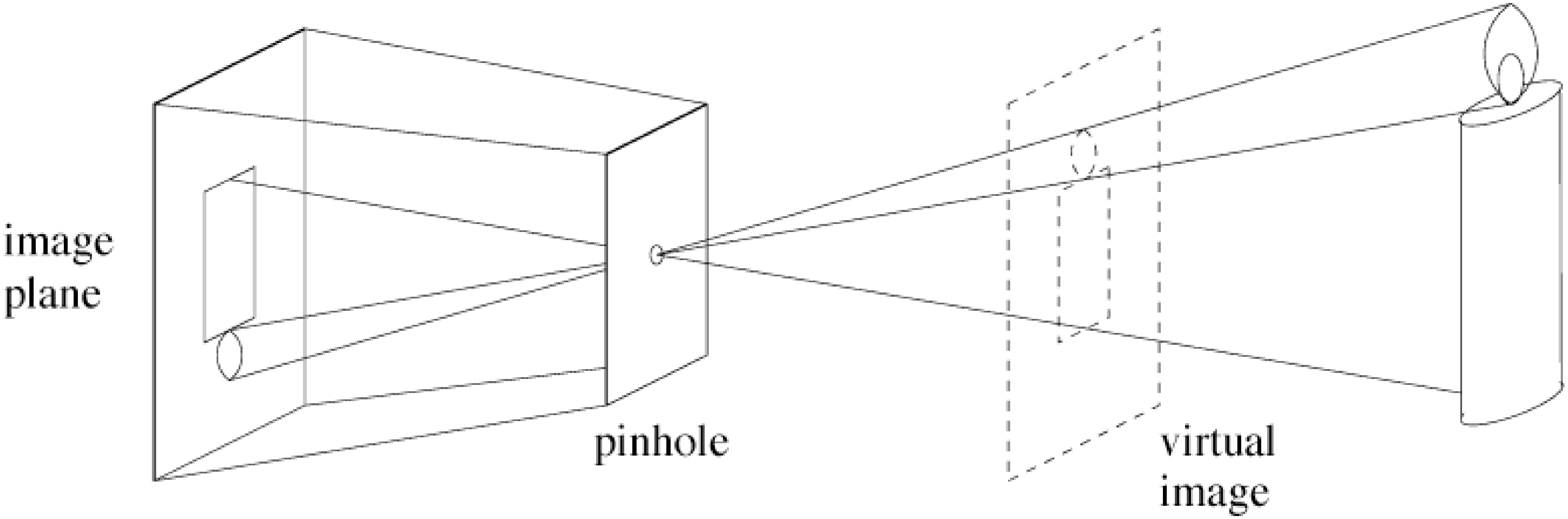
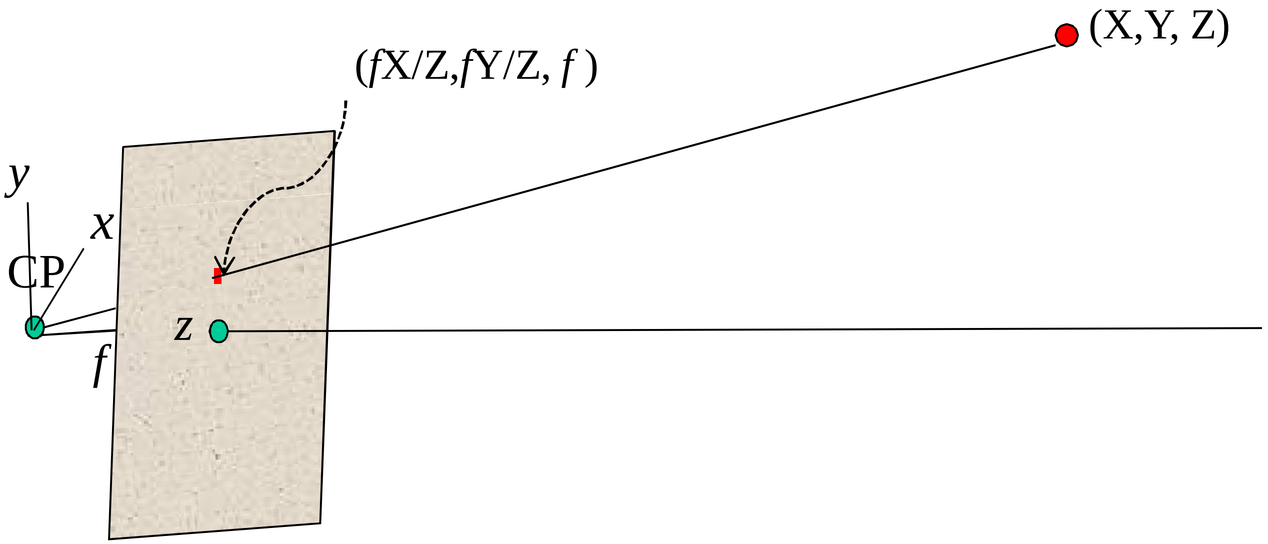


Image from D.A. Forsyth

Announcements

- HW #1 has been posted online and is due next Tue in class
- The TA office hours will be every Monday from 5-6:30pm at 301 Thomas.
- Any questions on previous lectures? Contact lhm@jpl.nasa.gov or one of the TA's
- Send UID to Shuo (hanshuo@caltech.edu) before midnight tonight to get building access.

Perspective Projection Revisited



$(X, Y, Z) \rightarrow (f\,X/Z, f\,Y/Z, f\,) \rightarrow (f\,X/Z, f\,Y/Z)$ in image

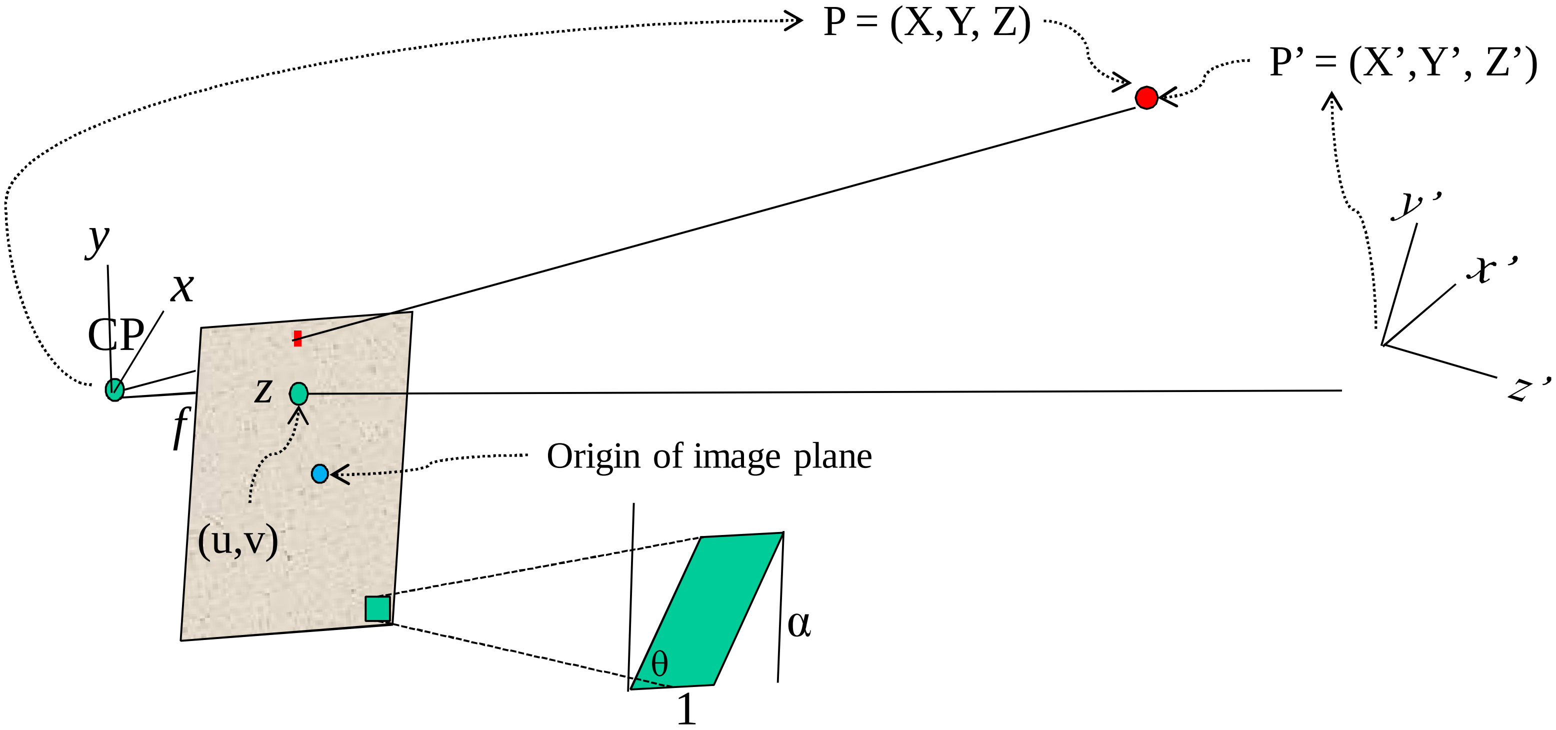
Note: $(\lambda X, \lambda Y, \lambda Z) \rightarrow (f\,X/Z, f\,Y/Z, f\,) \rightarrow (f\,X/Z, f\,Y/Z)$ in image

Implicit assumptions: (1) Only coordinate frame centered at CP.

(2) Origin of image plane = intersection point with z-axis.

(3) Image plane is uniform

Perspective Projection Generalized



- Let (R,T) = Euclidean transformation from global to camera frame s.t. R*P’ + T = P is expression for P’ in camera frame.
- (u,v) = coordinates of principal point (intersection of z axis of camera frame with image plane) in image plane.
- Plane has skew θ and aspect ratio α

Perspective Projection Generalized

- Projection from 3D to image plane works as follows:
- P’ = (X’, Y’, Z’) → P = (X, Y, Z) via P = R*P + T

- P = (X, Y, Z) → P_n = (X/Z, Y/Z) = normalized coordinates (*f* = 1 with idealized perspective projection)

- P_n → I = (x, y) = image coordinates via

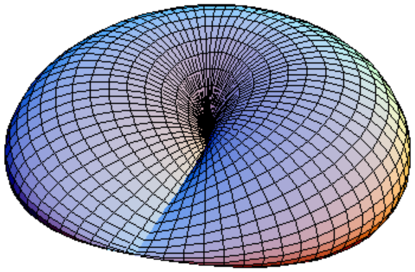
$$x = f\frac{X}{Z} + \sigma\frac{Y}{Z} + u$$

$$y = \alpha f\frac{Y}{Z} + v$$

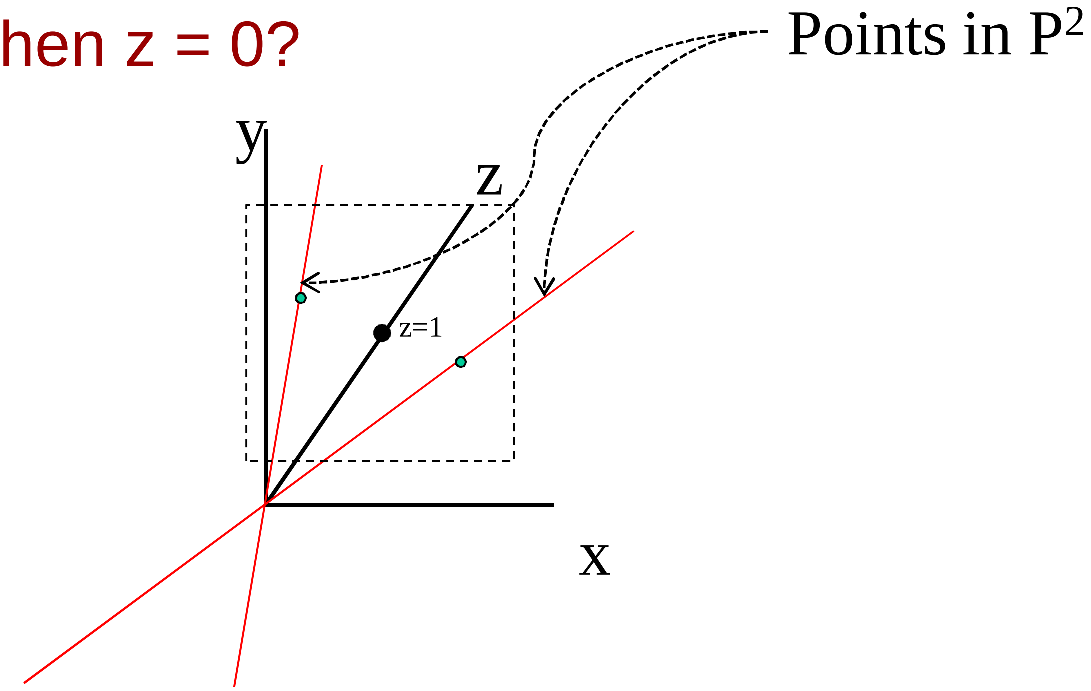
$$\sigma = f\cot(\theta)$$

Homogeneous Coordinates

- The Projective Plane P^2 is the set of equivalence class of points in $R^3 \backslash \{(0,0,0)\}$ subject to $p \approx \lambda p$ for λ in $R \backslash \{0\}$. Note that this describes all unoriented lines through the origin. Natural space to deal with perspective imaging.
- What happens when $z = 0$? E.g. $p = (1,1,0)$.



Cross-cap



Perspective Projection

- Advantage of homogeneous coordinates: If a point in the world is expressed in P^3 and a point in the image is expressed in P^2, there is a simple expression for image projection:

$$p = \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \begin{pmatrix} f & \sigma & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K_{3x3} E_{3x4} P = M_{3x4} P$$

- Observe P≈ (X,Y,Z,1), and p≈(x,y,1)

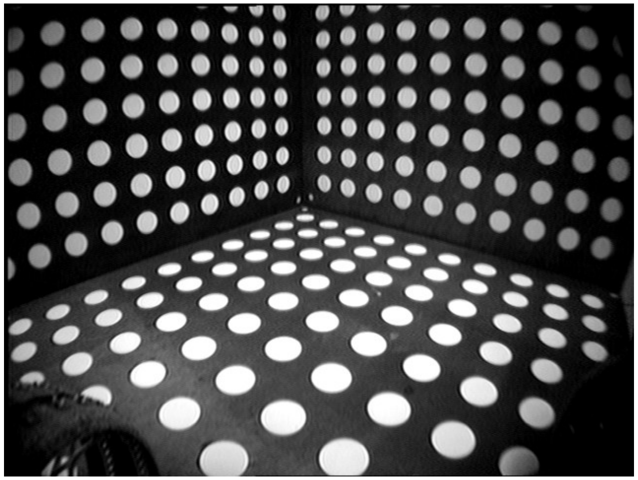
Intrinsics vs. Extrinsics

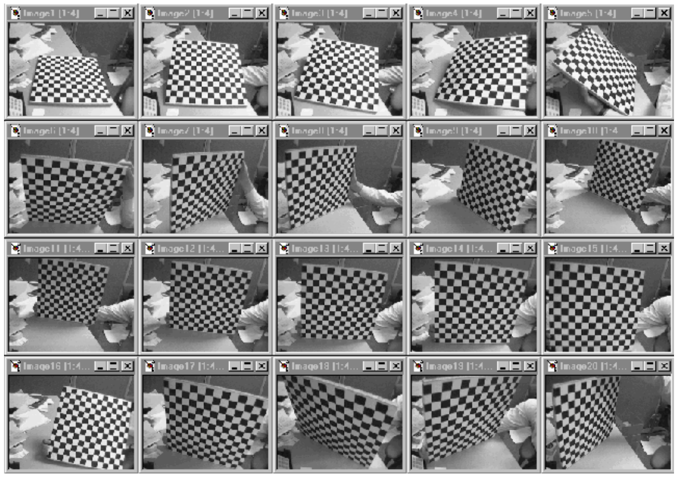
$$p = \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \begin{pmatrix} f & \sigma & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K_{3x3} E_{3x4} P = M_{3x4} P$$

- Extrinsics are the pose (position and orientation) of the camera in some pre-defined coordinate frame. Captured in Euclidean motion matrix E.
- Intrinsics (f, α, σ, u, v) describe the projection geometry of the camera independent of its pose (position and orientation). This is captured in the calibration matrix K.
- The product of the above is called the projection matrix (M in above notation). Given M, we can project any 3D point into image coordinates.
- Observe P≈ (X,Y,Z,1), and p≈(x,y,1)

What is Camera Calibration?

- Given enough correspondences of the form $\{P_i\} \leftarrow \rightarrow \{p_i\}$ between points in 3D and their corresponding image coordinates, compute M or better yet, K (intrinsics) and E (extrinsics).
- Typical input derived from pictures of a known object

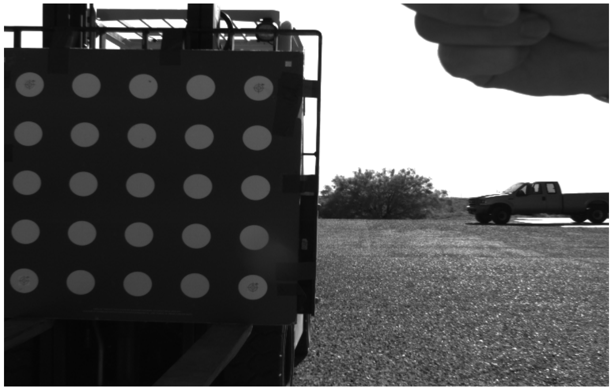


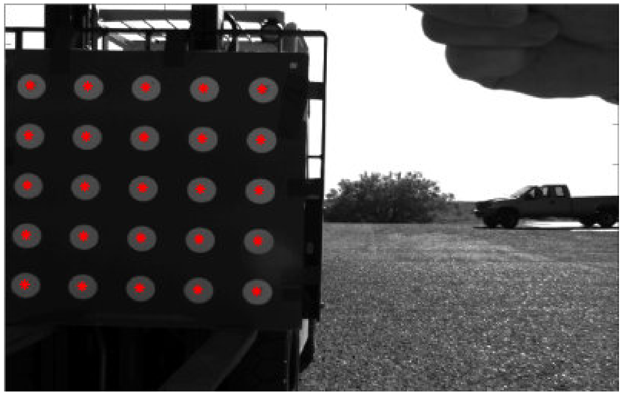


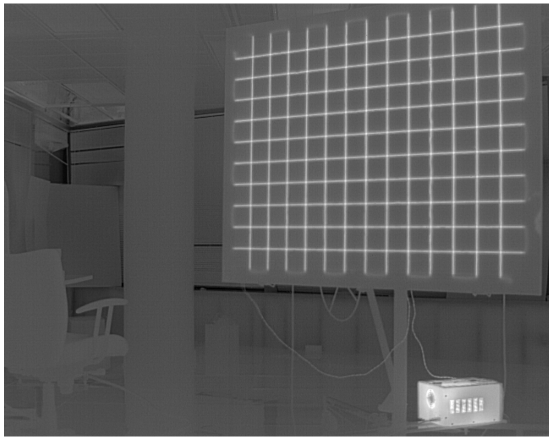
- 3D may be fully known or partially known

Input for Calibration

- Image coordinates $\{p_i\}$ derived from image processing algorithms: centroid finding, corner detection, etc.

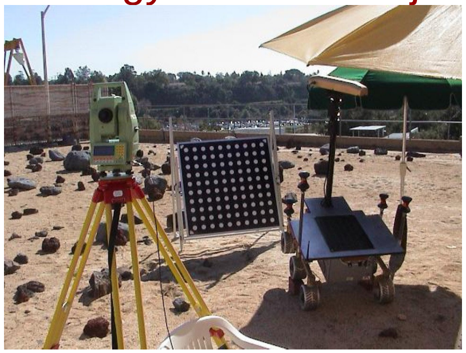


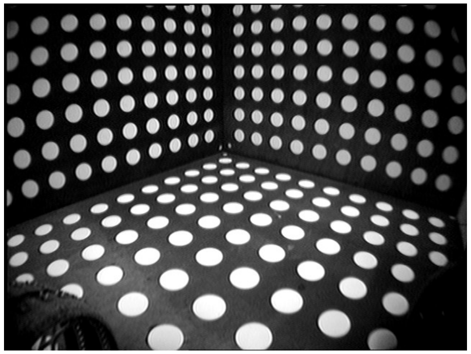




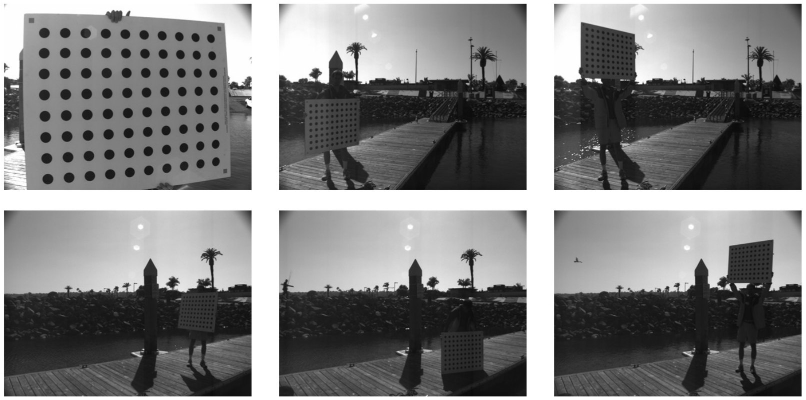
Input for Calibration

- 3D coordinates $\{P_i\}$ obtained from
  - Explicit metrology or known object geometry





  - For target planes, only per-plane geometry is known. Full 3D (i.e. orientation of plane in fixed coordinate frame is not known)



Linear Algebra Digression

- How do you minimize $||Ax||_2$ subject to $||x||_2=1$?
- Any m x n matrix A can be written in terms of its Singular Value decomposition

$$A = U_{m\times m} D_{m\times n} V^T_{n\times n}$$

Where the columns of U and V are orthonormal, and D is diagonal. The elements in D are called singular values of M. Columns of U corresponding to non-zero singular values span the range of M and columns of V corresponding to zero singular values span the null space of M.

(See Strang, *Linear Algebra and its Applications* or Golub and van Loan, *Matrix Computations*)

Linear Algebra Digression

$$A = U_{m\times m} D_{m\times n} V^{T}_{n\times n}$$

- What is the geometric intuition behind this?
- Suppose rank(A) = r < min(m,n). Then
  - V is a projection onto the co-kernel of A of dimension r.
  - D stretches along the right singular vectors by the singular values.
  - U expresses the result in the range of A in terms of the basis defined by the left singular vectors.
  - Imagine the effect on the unit sphere $||x||_2 = 1$ in R^n
    - Collapses into a r dimensional ellipsoid, where the “collapsing” directions correspond to the kernel of A
  - What if there are no zero singular values?
    - A is full rank, but direction of greatest collapse (i.e. the shortest axis of the ellipse) still corresponds to the best solution for minimizing $||Ax||_2$ subject to $||x||_2 = 1$

Linear Algebra Digression

- Note that for any matrix A, A^TA and AA^T are square, so we can compute their eigenvalue decompositions

$$A^T A = (UDV^T)^T (UDV^T) \quad = VDD^T V^T$$

- It follows that the non-zero singular values of A are the positive square roots of the non-zero eigenvalues of A^TA.
- Note the DD^T is a quadratic form. So we are back to ellipses and shortest axis. This is an alternate way to approach the problem.

Solving for the Projection Matrix, M

- Suppose we have a set of point correspondences $\{P_i\}$ ← → $\{p_i\}$ between 3D and 2D, where the full 3D is known in some coordinate frame. For each point, we have

$$\begin{pmatrix} \lambda_i x_i \\ \lambda_i y_i \\ \lambda_i \end{pmatrix} = M \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

$$x_i = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

$$y_i = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

A Standard Trick

- Rewrite the last set of equations as linear system in m_{ij}

$$A_{2n\times 12}\,\underline{m}_{12\times 1} = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1X_1 & -y_1X_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & X_2 & X_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ . \\ . \\ . \\ m_{33} \\ m_{34} \end{pmatrix} = 0_{6\times 1}$$

- For 6 non-coplanar points, matrix A has rank 11, hence 1-dimensional kernel. Let A = UDV^T be SVD. It follows that $\underline{m}$ is column of V corresponding to smallest singular value in D.

Solving for Intrinsics and Extrinsics

- Having solved for M up to scale, write it explicitly in terms of K, E

$$\begin{pmatrix} f & \sigma & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & T \end{pmatrix} = K \begin{pmatrix} R & T \end{pmatrix} = \begin{pmatrix} KR & KT \end{pmatrix} = \lambda M_{3\times 4}$$

- Let N = left 3x3 block of M, and S = last column of M
- Observe that $KR(KR)^T = KK^T = \lambda^2 NN^T$. Since $(KK^T)_{3,3}=1$, we can solve for λ up to a sign. Remaining solutions for K is straightforward algebra (you will do this for HW).
- Given K and λ, $R = \lambda K^{-1}N$. $T = \lambda K^{-1}S$.
  - Use SVD (orthonormal Procrustes) to refine R.

What if you only have planar targets?

- For each target, w.l.o.g., assume Z = 0.

$$\lambda\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K\begin{pmatrix} R & T \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longrightarrow \lambda\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K\begin{pmatrix} r_1 & r_2 & T \end{pmatrix}\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

- We can find the linear transformation H between the target plane and its image given at least 4 point.

$$H = \begin{pmatrix} h_1 & h_2 & h_3 \end{pmatrix} = \lambda K\begin{pmatrix} r_1 & r_2 & T \end{pmatrix}$$

- Since r$_1$ and r$_2$ are orthonormal:

$$h_1^T K^{-T} K^{-1} h_2 = 0$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2$$

What if you only have planar targets?

- We can express

$$h_1^T K^{-T} K^{-1} h_2 = 0$$
$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2$$

  as two linear equations in the elements of $K^{-T}K^{-1}$ (only 6 of these since the matrix is symmetric). This is analogous to what we did with M previously.

- 3 (or more) planes gives us 6 (or more) linear equations in $K^{-T}K^{-1}$. We solve exactly as with M.
- Going from $K^{-T}K^{-1}$ to K is straightforward algebra.
- $K^{-T}K^{-1}$ describes the Image Absolute Conic

Digression: CAHV

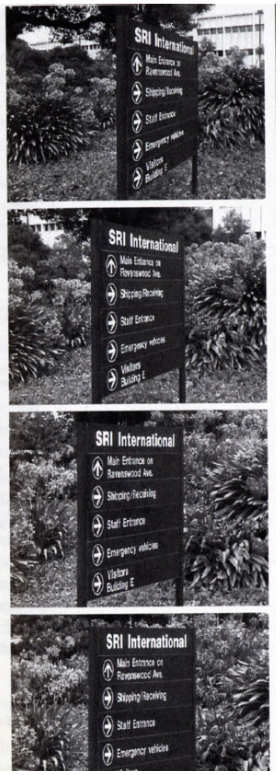
- Historically, a slightly different parameterization for perspective projection has been used at JPL. The camera is described by 4 3-dimensional vectors (C, A, H, V).
- P = (X,Y,Z) → p = (x,y) as follows:
- $x = H \circ (P-C) / A \circ (P-C)$
  $y = V \circ (P-C) / A \circ (P-C)$
- C = position of camera in global coordinate frame.
- A = pointing vector of camera (normalized)
- H = partially encodes rotation and partially encodes intrinsics
- V = partially encodes rotation and partially encodes intrinsics
- Note: 11 independent parameters above. For standard model, we have 5 (intrinsics) + 6 (extrinsic) = 11 parameters. These models are equivalent. Demonstrate this (HW).

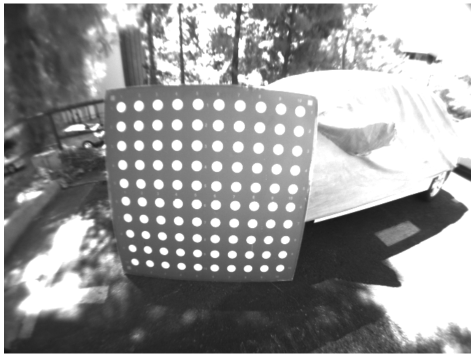
Real Cameras

Strong perspective:
Angles are not preserved
The projections of parallel lines intersect at one point

↓

Weak perspective:
Angles are better preserved
The projections of parallel lines are (almost) parallel





- But what about this?

Real Cameras

- Perspective projection is idealized. No real camera works like this.
  - Often a good approximation for narrow FOV → close to pinhole camera
- More sophisticated models are needed for more complex cases
  - Radial distortion is a radially symmetric function of distance from center of distortion.
  - Tangential distortion: arises from lens decentering, also commonly modeled
- We will apply non-linear distortion after Euclidean motion but before the camera matrix: i.e. in normalized coordinates. Any advantage to this ordering?
- Consider:

$$Q = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} = \begin{pmatrix} R & T \end{pmatrix} P$$

$$p_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} Q_x / Q_z \\ Q_y / Q_z \end{pmatrix}$$

Radial and Tangential Distortion

- Let (x_u,y_u) be the result of applying radial and tangential corrections to (x_n,y_n)

$$x_u = x_n + x_n(k_1r^2 + k_2r^4 + k_3r^6) + 2p_1x_ny_n + p_2(r^2 + 2x_n^2)$$
$$y_u = y_n + y_n(k_1r^2 + k_2r^4 + k_3r^6) + 2p_2x_ny_n + p_1(r^2 + 2y_n^2)$$

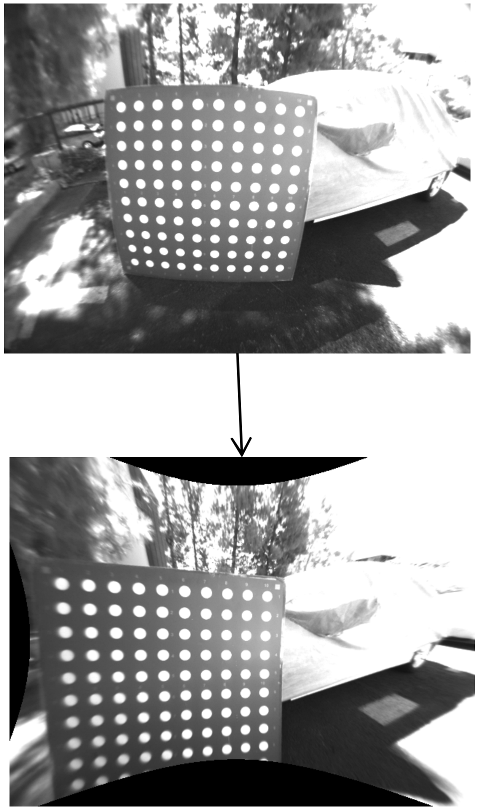
- Then the image coordinates corresponding to P are given by

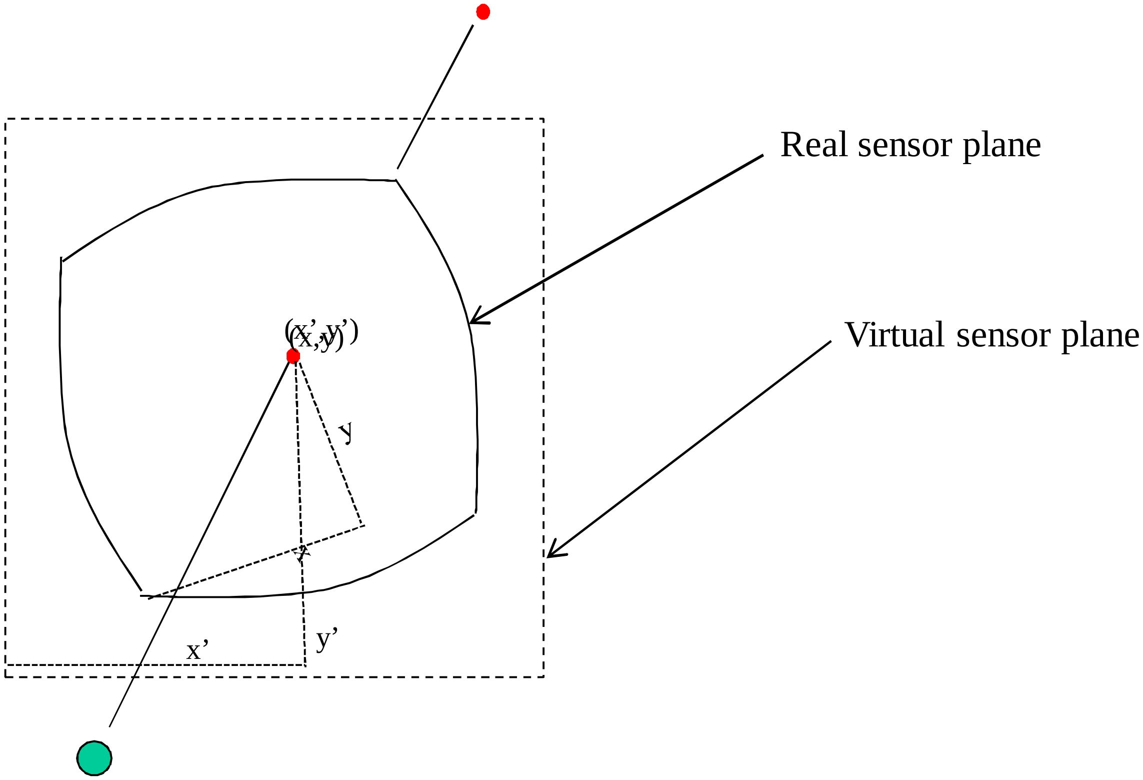
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix}$$

Where K is the calibration matrix discussed above.

“Linearizing” and Image

- Given an image with non-linear distortion, we generate a virtual image that is pure perspective. Imagine a virtual perspective camera with projection center identical to the real camera’s. What would this virtual camera see?





Digression: CAHVOR

- The standarn non-linear model used at JPL is called CAHVOR. It adds two additional 3-vectors to the linear CAHV model.
- O = axis of distortion in 3D
- R = parameters of polynomial distortion

- Observe that this is neither radial nor tangential distortion. However, it captures the situation where the nominal optical axis of the lens is not parallel to the image plane.

- Further extension: CAHVOR(E)
  - Moving center of projection.
  - Explicit blending of fisheye + perspective models
  - Used for Hazcams on Mars rovers.

How to Solve for non-linear models

- Possible to decouple radial distortion from linear model and solve for linear first. How?
- In general, this is not how we solve for arbitrary models. Instead, use non-linear optimization techniques.
- Consider the full transfer function F between 3D and image coordinates. The parameterization can be arbitrary and can include both extrinsics and intrinsics.

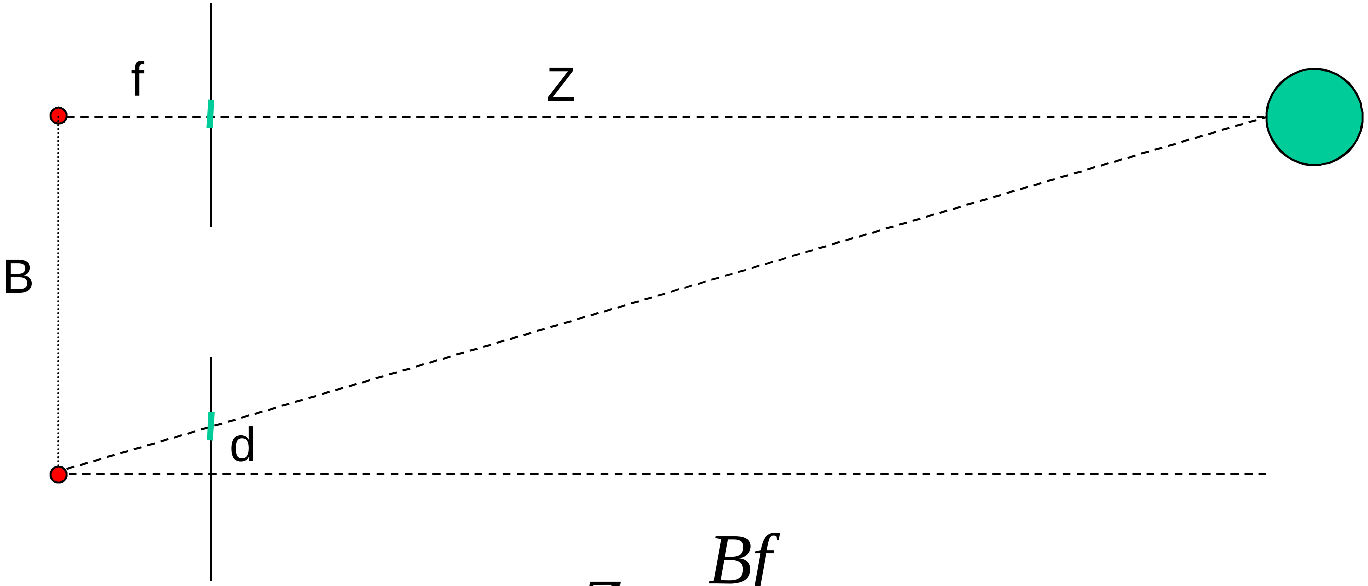
$$F(\phi_i)(P) = p$$

where {Φ_i} ranges over all parameters in the model. Then calibration can be phrased as

$$\bar{\phi}_i = \arg\min_{\phi_i} \sum_j \left\| F(\phi_i)P_j - p_j \right\|^2$$

Introduction to Depth from Stereo (1-D)

- Given a pair of calibrated cameras with known relative pose, we can compute the range to 3D points by triangulation.

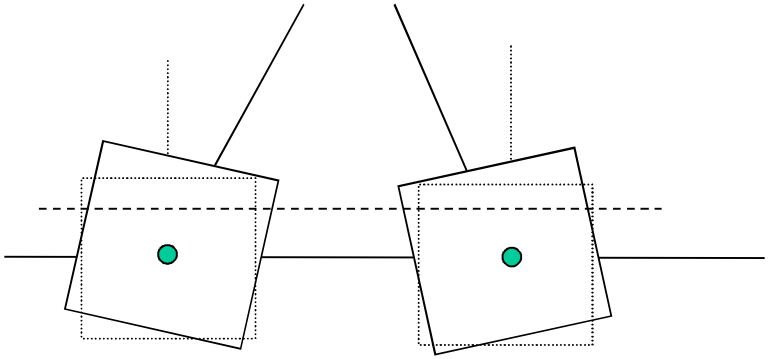


$$Z = \frac{Bf}{d}$$

Must solve correspondence problem.

Stereo in 2-D

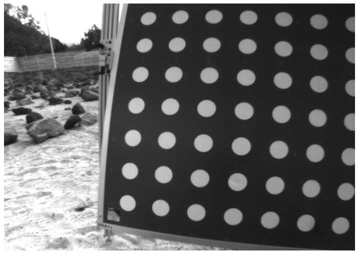
- Straightforward extension from 1-D geometry to 2-D
- However:
  - Solving correspondence problem in 2-D is expensive
  - Camera alignment and image rectification
    - Turns 2-D search into 1-D search

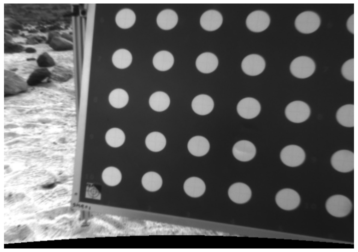


Geometry recovered through camera calibration

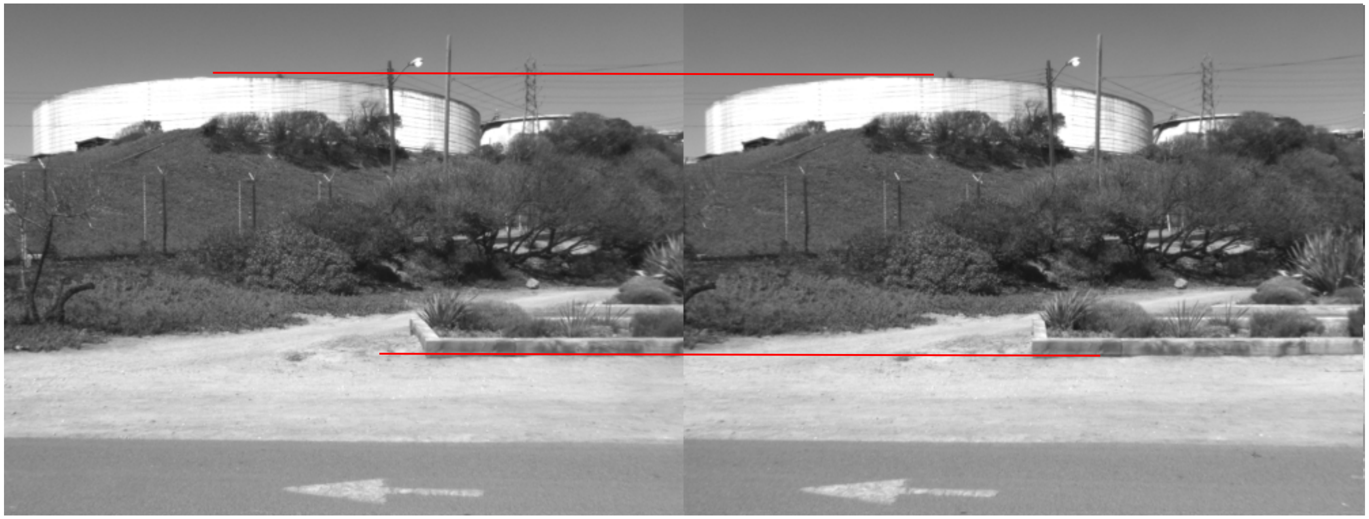
Image rectification

- Project data from real imagery to aligned virtual cameras.
  - Remove image non-linearity



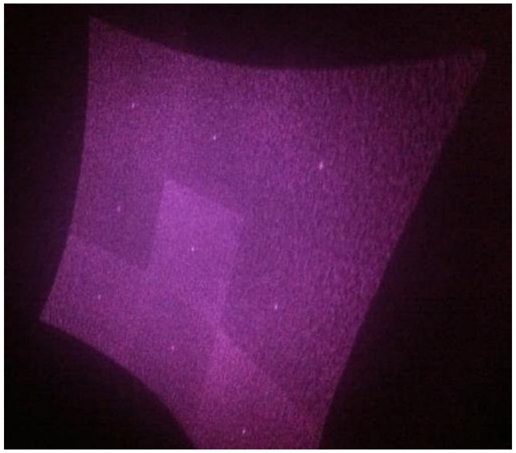


  - Modify intrinsics (focal length, image center) and camera pointing



How Does a Kinect Work?

- Principal is identical to binocular stereo except:
  - Projected pattern of light is matched to reference pattern
- Still a line search problem
- Still need to calibrate
  - How?



How to Rectify a Stereo Pair

- Virtual cameras must share projection centers with real cameras.
- Virtual cameras should have identical intrinsics (to each other), no non-linear distortion
- Rotate image planes about projection centers by introducing new coordinate frame as follows:
  - e1 parallel to baseline vector
  - e2 average of normals to sensor plane
  - e3 = e2 x e1
  - Compute rotation matrices R_l and R_r for left and right cameras that take x → e1, z → e2, y → e3

Extra Considerations in Calibrating Stereo

- We can calibrate individual cameras separately, provided 3D data is in a consistent coordinate frame across cameras.
- This ignores some strong constraints.
- Extra constraints:
  - Triangulate and reproject
  - 3D scene structure is estimated during calibration

Performing a real calibration

- Lots of calibration packages out there. We will use Matlab Calibration Toolbox (J-Y Bouguet) as reference.
- Mapping our notation to Bouguet’s
  - *f* ← → *fc(1)*
  - α *f* ← → *fc(2)*
  - u ← → cc(1)
  - v ← → cc(2)
  - σ ← →alpha_c
  - k1, k2, k3 ← →kc(1), kc(2),kc(5)
  - p1,p2 ← → kc(3), kc(4)

Performing a real calibration

<Example from Matlab Toolbox>

Homework

- Read Szeliski 2.1.
- Familiarize yourself with: http://www.vision.caltech.edu/bouguetj/calib_doc/
- Problems (due 1/24/11):

1. Assuming a camera with standard linear model plus radial and tangential distortion, formally compute the effect on image projection of distorting f, k_1 and u. Assume only one parameter changes at a time.
2. Use the Matlab Calibration Toolbox to calibrate the Zhang dataset. For each of the parameter f, k_1 and u, plot the error induced in the projection (you can use project_points2.m) of the point (0,0,1) as a function of small changes in the respective parameter. Does this behavior match your prediction in Q.1 above?
3. Implement the linear solution of the projection matrix, calibration matrix and pose as described in class. Calibrate the data in CalibrationData.mat using your solution. Now calibrate it using the Matlab Toolbox (Refer to 3rd calibration example, demo_script_oulu_3D.m). Do your results match?

References

- R. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses,” IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, Aug. 1987.
- Z. Zhang, “Flexible Camera Calibration by Viewing a Plane from Unknown Orientations,” ICCV 99.
- D. Gennery, “Generalized Camera Calibration Including Fish-Eye Lenses”, IJCV 2006.