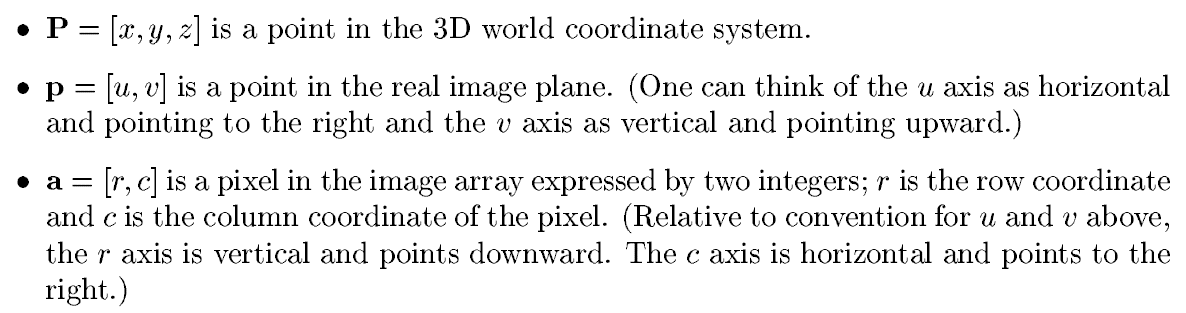
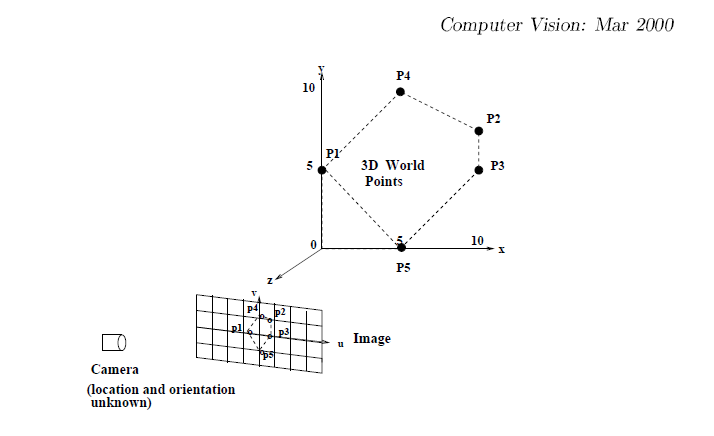
**Camera Calibration**

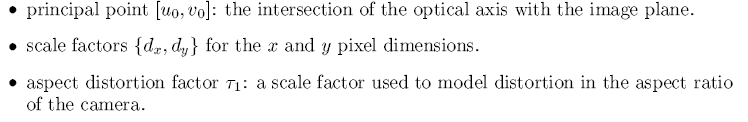
1. Definition of Coordinates:



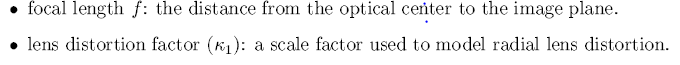
P is defined in the world frame and p is defined in the image/camera frame. Here is a picture for reference:



1. Intrinsic Parameters: (Specific to every Device)
2. What we need to know:



1. What we can calculate during the calibration process:

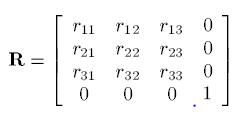


Note: The Camera origin is at the optical center of the camera lens.

Extrinsic Parameters: (Depends on camera orientation and reference point)

These parameters describe the position and orientation of the camera system in the 3D world. They include:

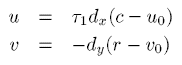
1. Translation: t = [tx ­ty tz]T
2. Rotation:



There are only three independent rotation parameters which are the three angles not nine.

Procedure:

1. Do not focus or move the camera during calibration
2. We must know more than 5 correspondences between real world points Pi = (xi,yi,zi) and their real world image coordinates. pi = (ui, vi). For each correspondence the distance to the screen can be varied. {([xi,yi,zi], [ui, vi]) | i = 1,….,n} n>5. ui and vi have origin at center of image
3. Real world image co-ordinates can be calculated by the formula:

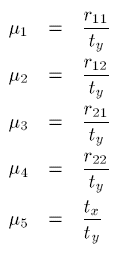
where r,c are array indices of pixels. R is row coordinate and c is column coordinate

1. Create a Matrix A with rows ai :



1. Define vector:

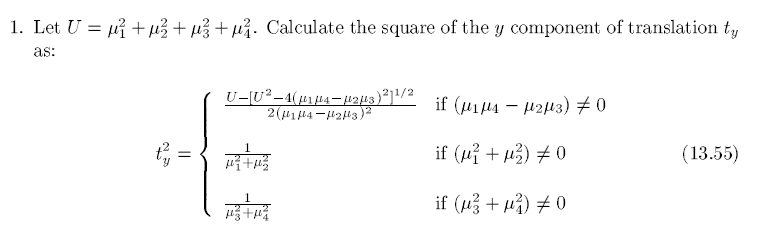






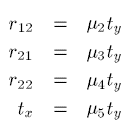
1. We Know A and b, now we can find mu with formula:

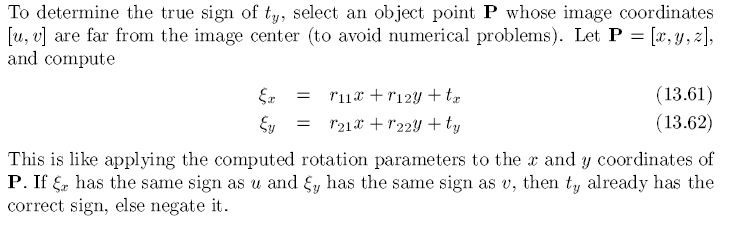
Solving tz comes later because change in tz has an image effect similar to Kappa 1.

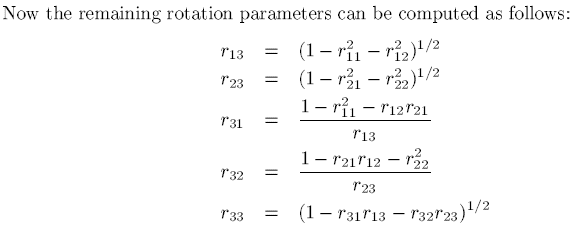


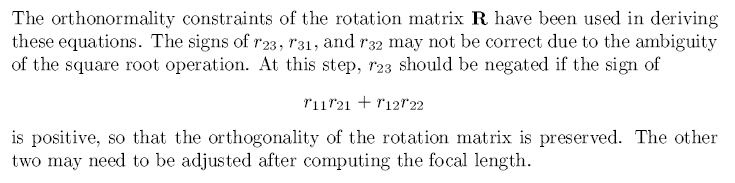
1. Now calculate rotation matrix parameters and tx taking positive square root of ty:



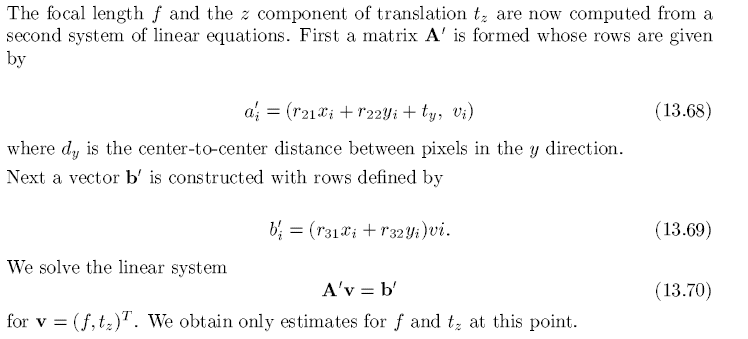






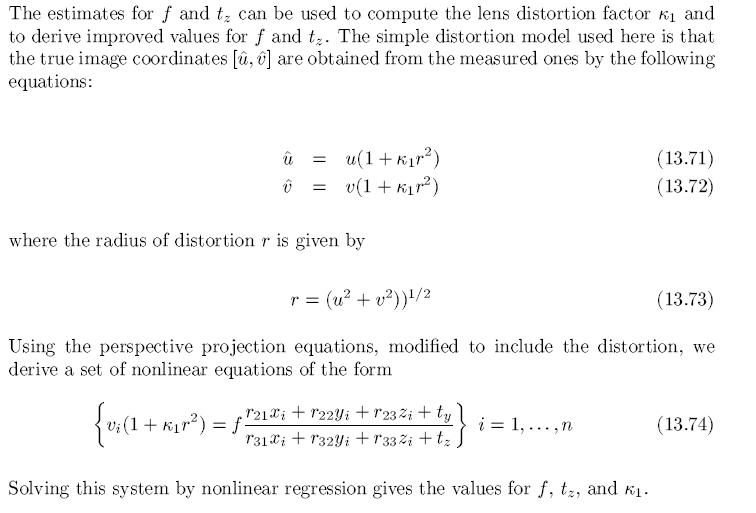


12)



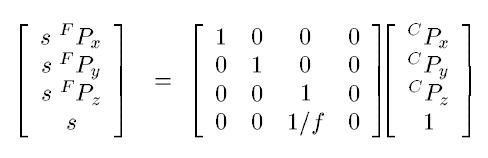


1. This is an extra step to increase accuracy:

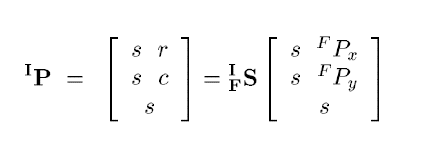


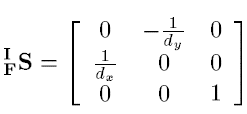
I think we can use these new kappa and f to find the mapping again and get a more accurate one.

1. After getting transform cTw invert it to get screen/world in camera co-ordinate system.
2. From 3D camera-coordinates we can get 2D real world image distances:



1. Note that FPis the same as (u,v) but instead the center is at the bottom left of the screen, note that the z coordinate is always the focal length and s is a constant.
2. From FP we can get IP which are pixel coordinates, these coordinates have origin at top left:





1. Multiplying all the transforms we get a transform to go from 3D world coordinates to the pixel plane. We can find pseudo inverse of this to get pixel plane to 3D world co-ordinates.