Calibration Overview

Jason Rebello

10/07/2017

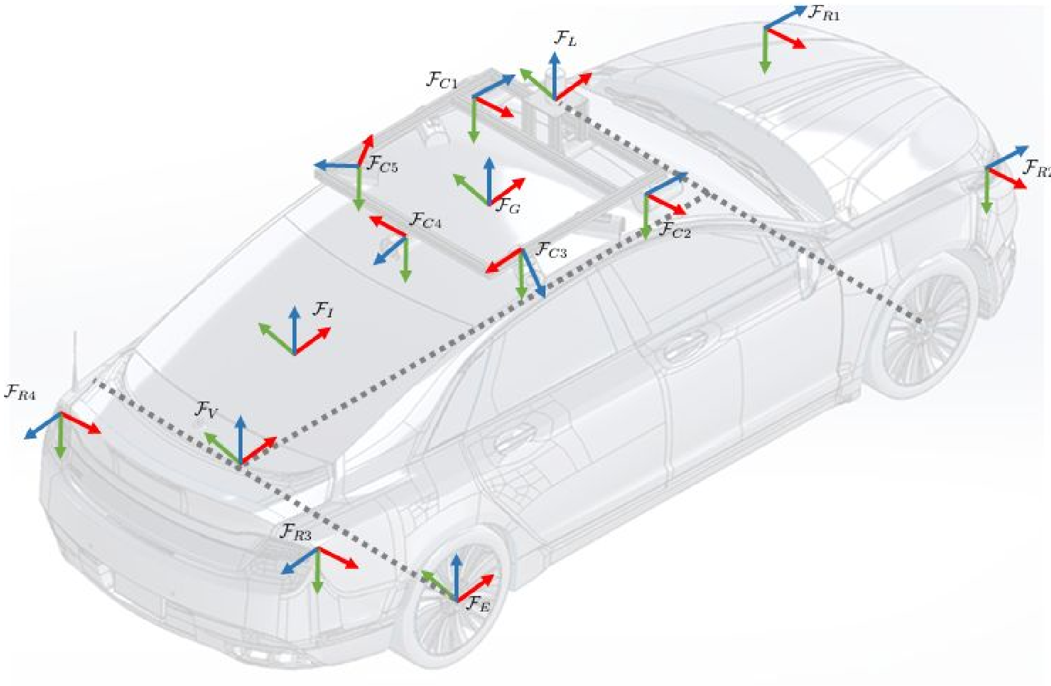
No moose was hurt in the making of this presentation

Calibration | Why do we need calibration ?

- Data from multiple complementary sensors leads to increased accuracy and robustness

- Sensors need to be spatially and temporally calibrated

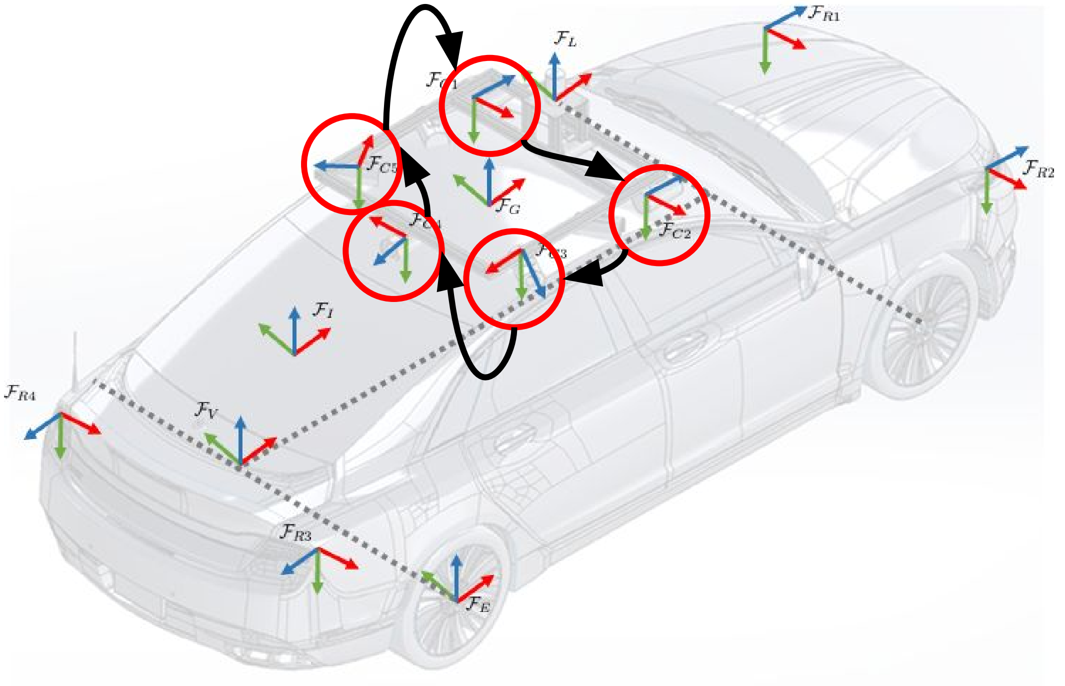
- Calibration helps transform measurements from different sensors into a common reference frame



Source: Arun Das

Required Calibrations

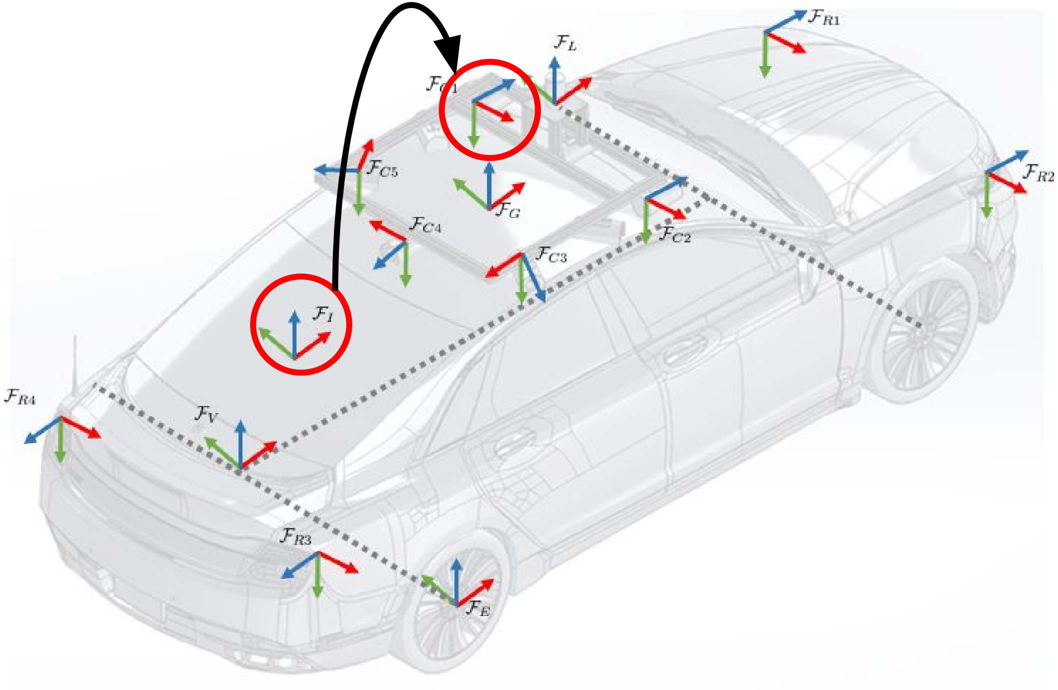
- Camera intrinsic and extrinsic
- IMU to Camera
- Lidar to GPS
- Lidar to Camera



Source: Arun Das

Required Calibrations

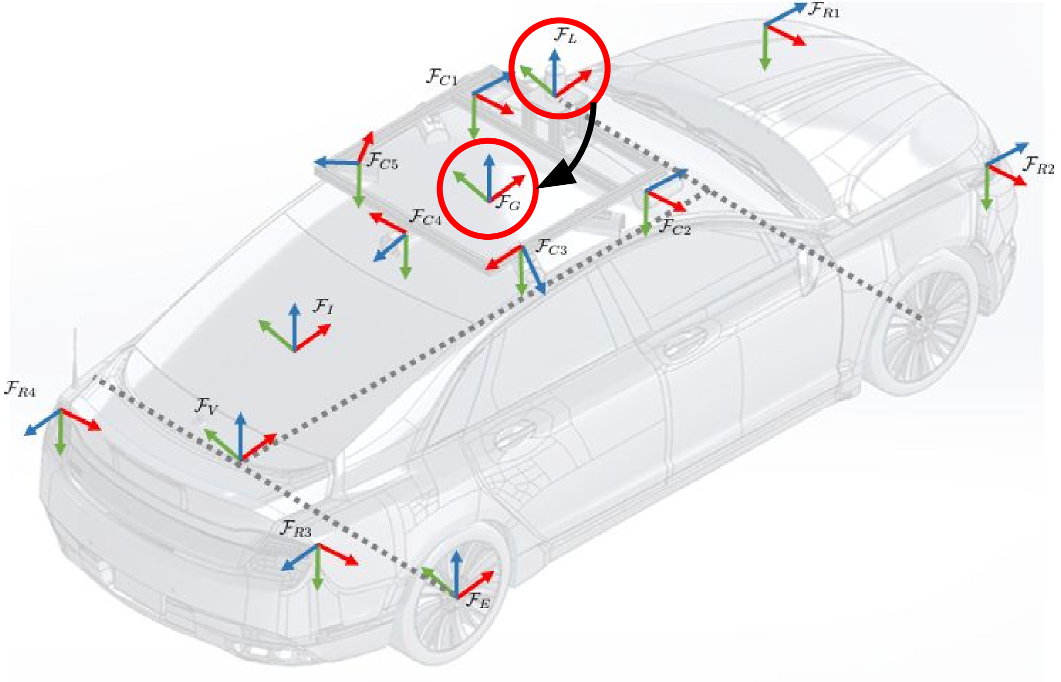
- Camera intrinsic and extrinsic
- IMU to Camera
- Lidar to GPS
- Lidar to Camera



Source: Arun Das

Required Calibrations

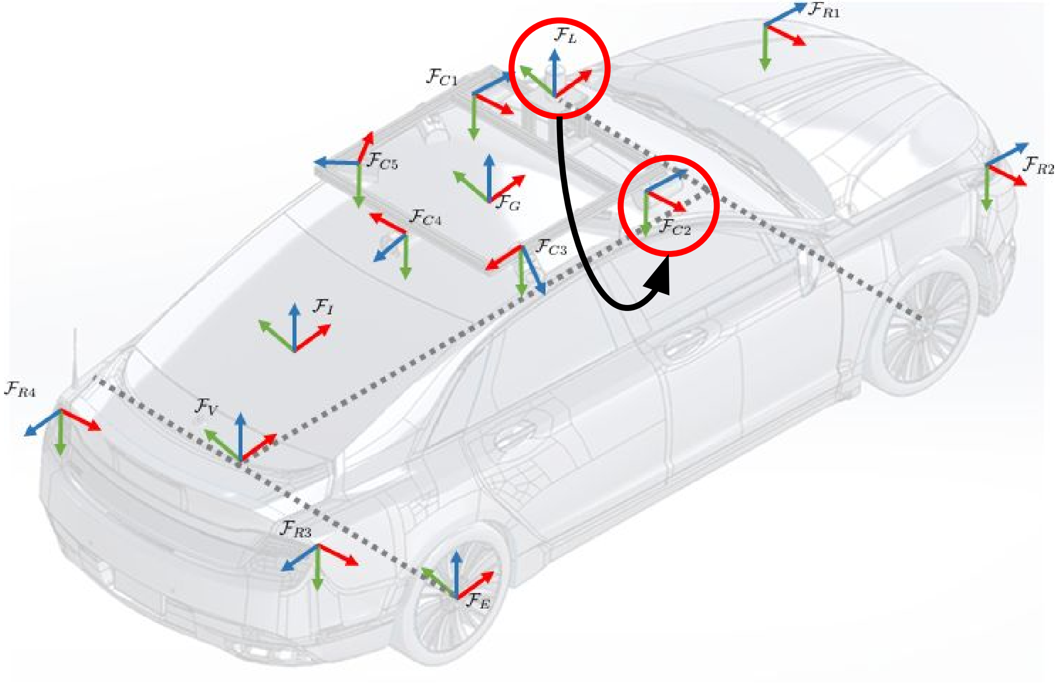
- Camera intrinsic and extrinsic
- IMU to Camera
- Lidar to GPS
- Lidar to Camera



Source: Arun Das

Required Calibrations

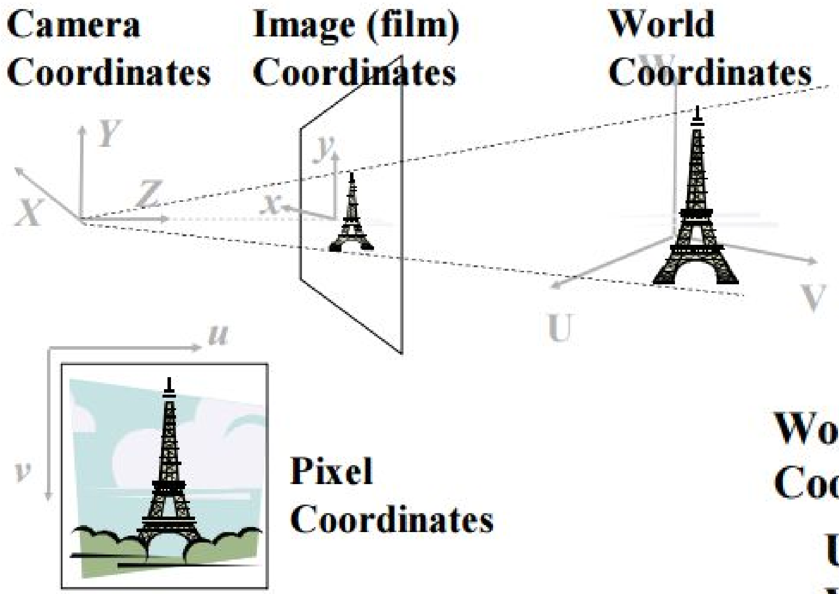
- Camera intrinsic and extrinsic
- IMU to Camera
- Lidar to GPS
- Lidar to Camera



Source: Arun Das

Camera Intrinsics

Forward Projection



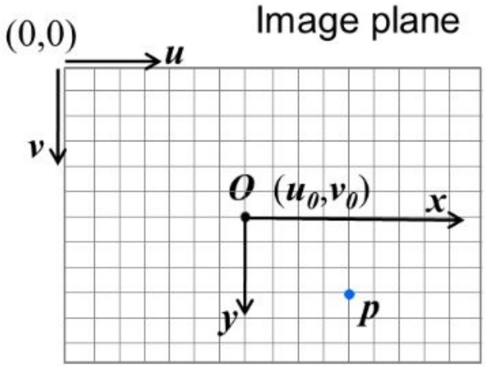
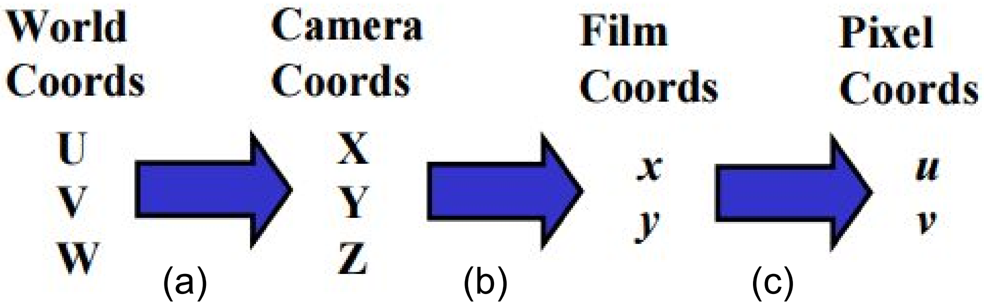


Image to Pixel Coordinates

Extrinsics R,t

Intrinsics focal length, ox, oy



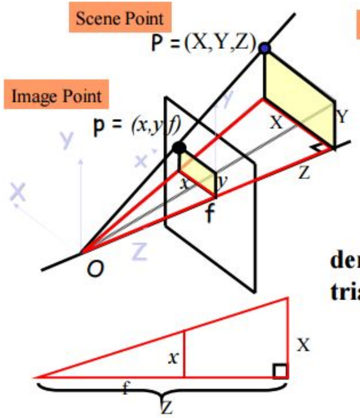
Source: Robert Collins, CSE486

(a) Extrinsic Transformation (Rotation + Translation)

- Transforms points from World to Camera coordinate Frame
- Homogeneous coordinates allow for easy matrix multiplication

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

(b),(c) Perspective Projection



Perspective Projection Eqns

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

derived via similar triangles rule

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f/s_x & 0 & o_x & 0 \\ 0 & f/s_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera Coordinates

$$u = \frac{x'}{z'}$$

$$v = \frac{y'}{z'}$$

Pixel Coordinates

Source: Robert Collins, CSE486

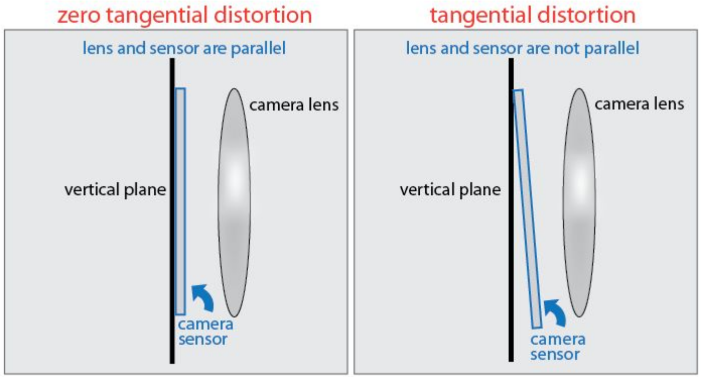
No distortion

Barrel distortion

Pincushion

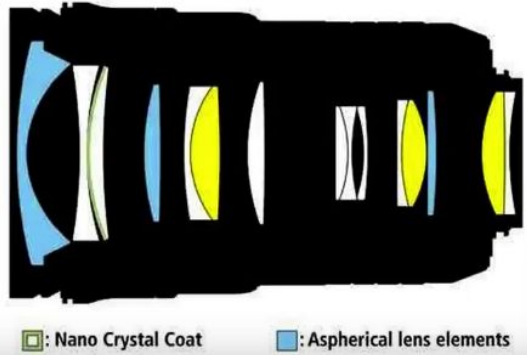
Radial Distortion

Source: Scaramuzza



Tangential Distortion

Source: MathWorks



Nikon Lens System

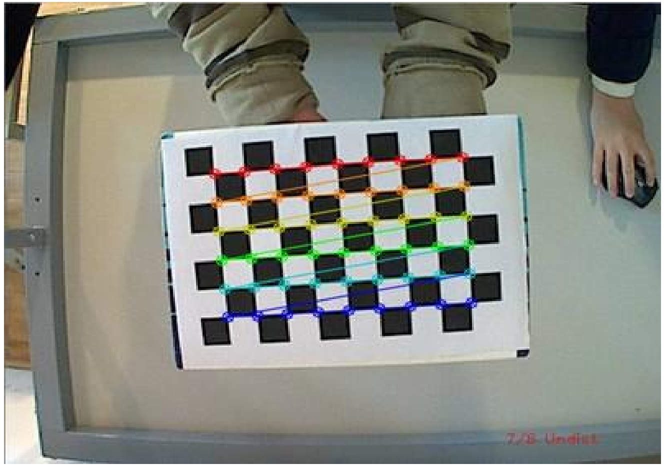
Source: Aaron Bobik

Camera Calibration using a 2D Checkerboard

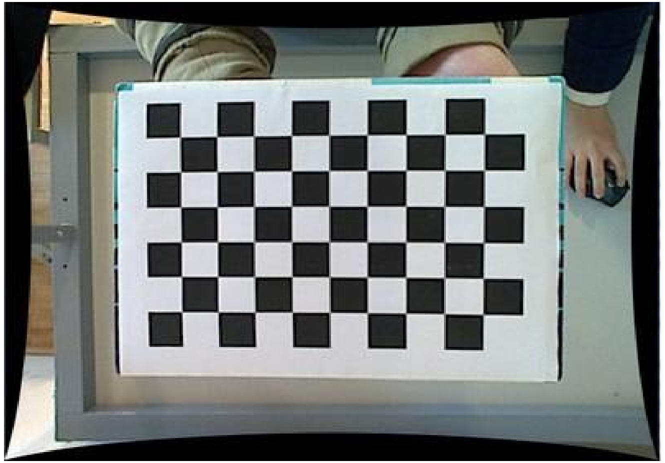
Known size (30 mm) and structure (6 x 9)

Identify corners easily.

Distorted



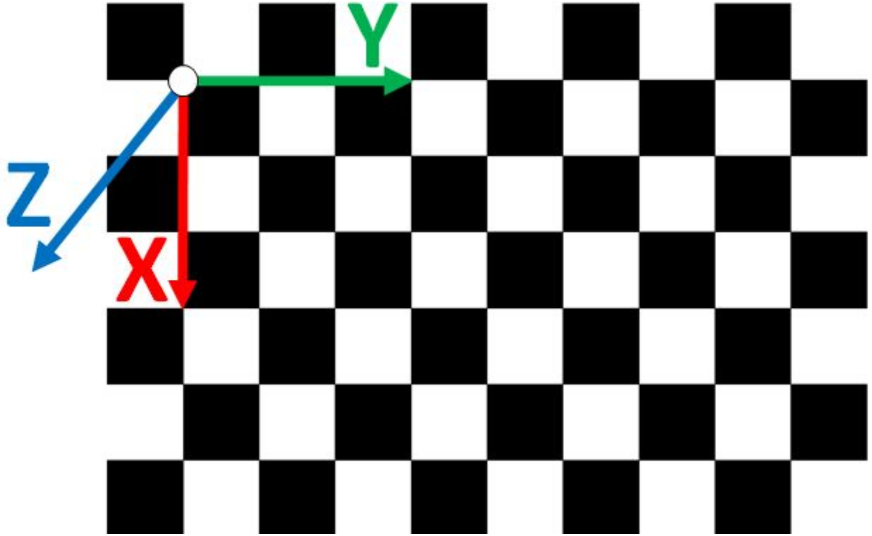
Corrected



Where do I set my world coordinate frame ?

Can set the world coordinate system to one corner of the checkerboard

All points lie in the X,Y plane with Z=0



Projection |

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Intrinsic camera parameters

Extrinsic camera parameters

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cancel{r_{13}} & t_1 \\ r_{21} & r_{22} & \cancel{r_{23}} & t_2 \\ r_{31} & r_{32} & \cancel{r_{33}} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ \cancel{0} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \overbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}}^{\boxed{\text{Homography } \mathbf{H}}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3) = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}}_{(\mathbf{r}_1, \mathbf{r}_2, \mathbf{t})}$$

$$(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3) = \mathbf{K}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{t})$$

$$\Rightarrow \quad \mathbf{r}_1 = \mathbf{K}^{-1}\mathbf{h}_1, \qquad \mathbf{r}_2 = \mathbf{K}^{-1}\mathbf{h}_2$$

$$H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$$

$$-h_1x - h_2y - h_3 + (h_7x + h_8y + h_9)u = 0$$

$$-h_4x - h_5y - h_6 + (h_7x + h_8y + h_9)u = 0$$

$$A_i\mathbf{h} = \mathbf{0}$$

$$A_i = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & v \end{pmatrix}$$

$$\mathbf{h} = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{pmatrix}^T$$

$$\mathbf{r}_1^T \mathbf{r}_2 = 0, \qquad \|r_1\| = \|r_2\| = 1$$

$\mathbf{r}_1^T \mathbf{r}_2 = 0$

$$\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 = 0$$

$\|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1$

$$\mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2$$

$$\mathbf{B} := \mathbf{K}^{-T}\mathbf{K}^{-1}$$

Find B , recover K through the Cholesky decomposition chol (B) = AA^T

A = K^{-T}

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{pmatrix}$$

$$\mathbf{b} = (b_{11}, b_{12}, b_{13}, b_{22}, b_{23}, b_{33})$$

Construct a system of linear Equations Vb = 0

$$\mathbf{V} = [\ v^T_{12} \quad v^T_{11} - v^T_{22}\ ]^T$$ … 2x6 matrix for 1 image

Where

$$v_{ij} = [\ h_{1i}h_{1j}\ ,\ h_{1i}h_{2j}+h_{2i}h_{1j}\ ,\ h_{3i}h_{1j}+h_{1i}h_{3j}\ ,\ h_{2i}h_{2j}\ ,\ h_{3i}h_{2j}+h_{2i}h_{3j}\ ,\ h_{3i}h_{3j}\ ]$$

For n images

V = 2n x 6 matrix

Each plane gives us two equations

Since B has 6 degrees of freedom, we need at least 3 different views of a plane

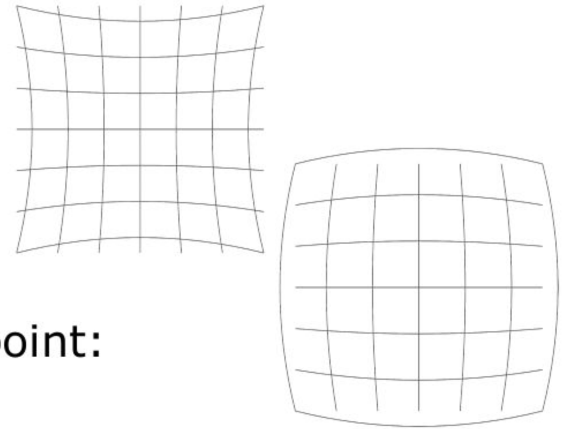
We need at least 4 points per plane

Real measurements are corrupted with noise

Find a solution that minimizes the least-squares error

$$b = \arg\min_{b} \mathbf{Vb}$$

Lens distortion



Non-linear effects:

- Radial distortion
- Tangential distortion

- Compute corrected image point:

$$(1)\quad \begin{aligned} x' &= x/z \\ y' &= y/z \end{aligned}$$

$$(2)\quad \begin{aligned} x'' &= x'(1+k_1r^2+k_2r^4)+2p_1x'y'+p_2(r^2+2x'^2) \\ y'' &= y'(1+k_1r^2+k_2r^4)+p_1(r^2+2y'^2)+2p_2x'y' \end{aligned}$$

where $r^2 = x'^2 + y'^2$ $\quad k_1, k_2$: radial distortion coefficients

p_1, p_2 : tangential distortion coefficients

$$(3)\quad \begin{aligned} u &= f_x \cdot x'' + c_x \\ v &= f_y \cdot y'' + c_y \end{aligned}$$

Lens distortion can be calculated by minimizing a non-linear function

$$\min_{(\mathbf{K},\kappa,\mathbf{R}_i,\mathbf{t}_i)} \sum_i \sum_j \|\mathbf{x}_{ij} - \hat{x}(\mathbf{K}, \kappa, \mathbf{R}_i, \mathbf{t}_i; \mathbf{X}_{ij})\|^2$$

Estimation of κ using non-linear optimization techniques (e.g. Levenberg-Marquardt)

The parameters obtained by the linear function are used as starting values

Before calibration:

After calibration:

1) Print a target checkerboard (Several available online)
2) Note dimensions of target, size of square
3) Take multiple images by moving target to different locations on the image
4) Calibrate using OpenCV, Matlab
5) Reprojection error below 0.2 is good

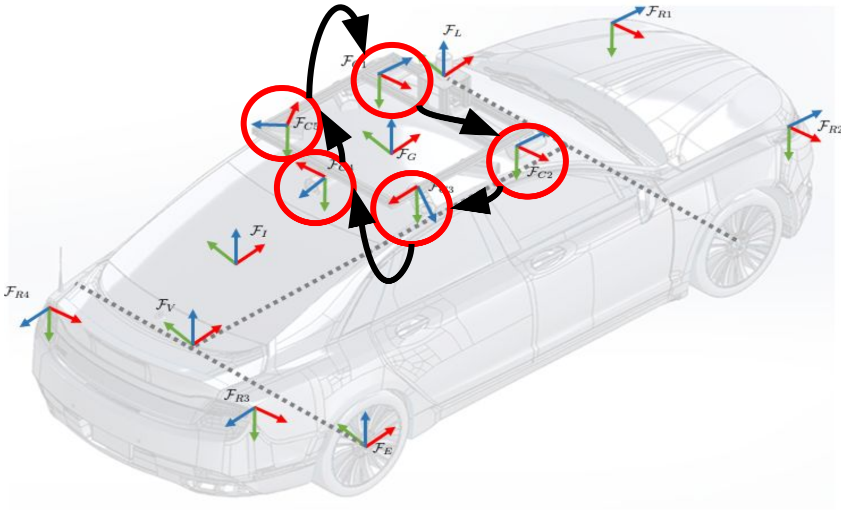
```
image_width: 900
image_height: 600
camera_name: narrow_stereo
camera_matrix:
  rows: 3
  cols: 3

data: [657.603916, 0.000000, 449.430625,
       0.000000, 657.439333, 326.379426,
       0.000000, 0.000000, 1.000000]

distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.210857, 0.077466, -0.000022, -0.000545, 0.000000]
```

Ximea camera: 5.3 μm / pix
Edmund Optic lens: 3.5 mm

Focal length in pixel = $3.5 / (5.3 \times 10^{-3}) = 660.37$

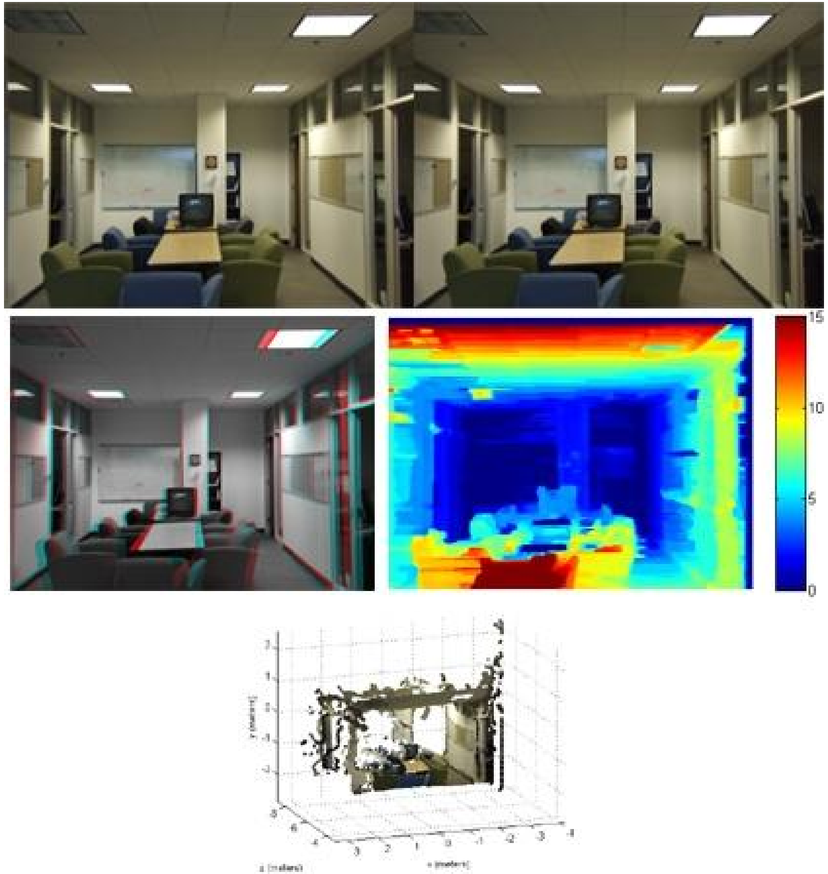
Question: What is the problem with this sort of calibration ?

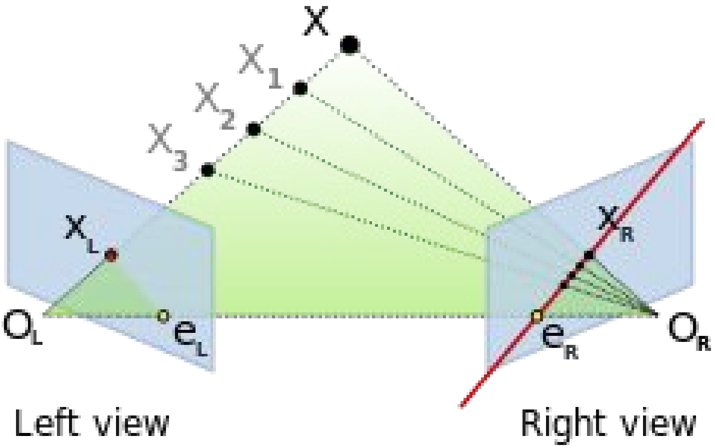
Camera Extrinsics



Rotation and Translation

Advantages of Stereo Vision |





Stereo Vision:

1) Recover Depth

2) Limit the search space in the second image for point correspondence

Given:

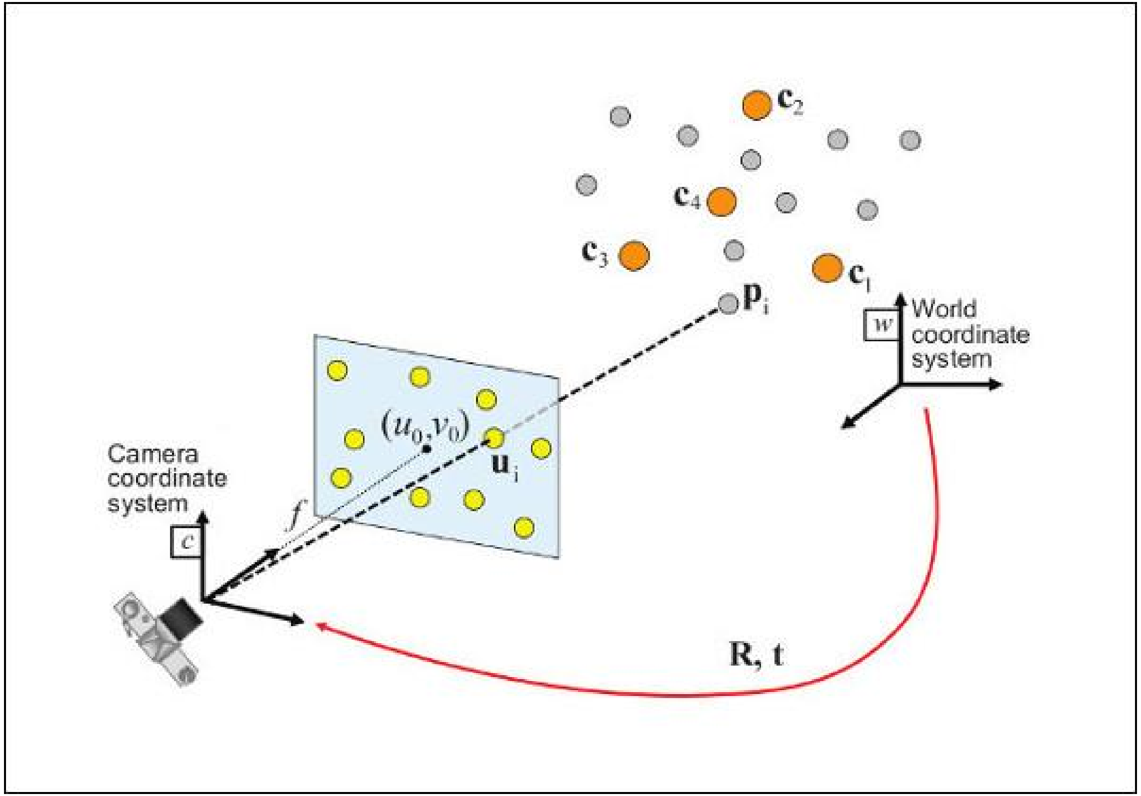
1) Known 3D points in world frame (Checkerboard Points)

2) Pixel Locations on image plane

3) Intrinsic parameters of Camera

Estimate:

Transformation from world to camera



1) Given 3D points in world frame P^w and corresponding 2D pixel locations in both cameras z_s and z_m

2) Estimate Transformations from world to both cameras using PnP algorithm $T^{s:w}$ and $T^{m:w}$

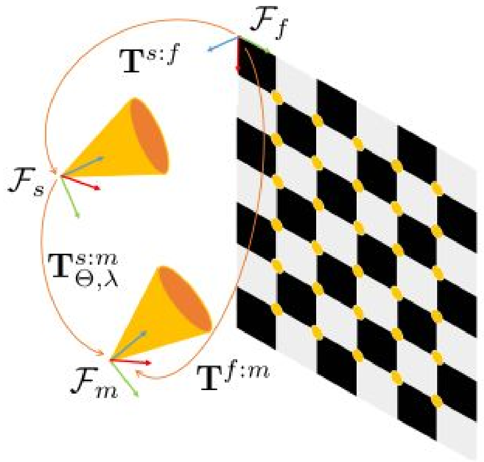
3) Transform 3D points from world to F_s camera.
$P^s = T^{s:w} P^w$

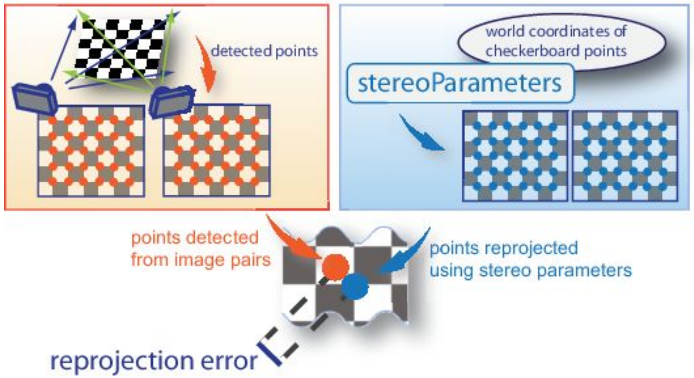
4) Transform 3D points from camera F_s to camera F_m via Extrinsic transformation.
$P^m = T^{m:s} P^s$

5) Project points onto F_m. camera. $z'_m = \pi(P^m)$

Reprojection Error:

$$\sum_{images} \sum_{points} d(z'_m, z_m)^2 + d(z'_s, z_s)^2$$





UNIVERSITY OF WATERLOO

How should I calibrate the extrinsics ? |

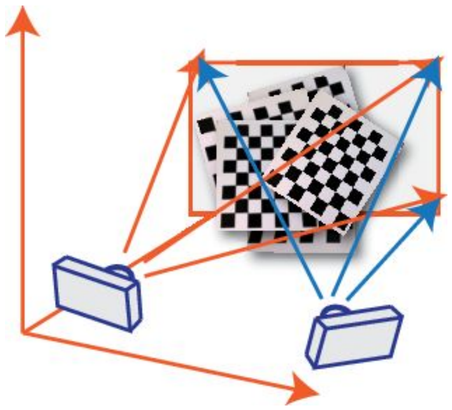
1) Print a target checkerboard (Several available online)

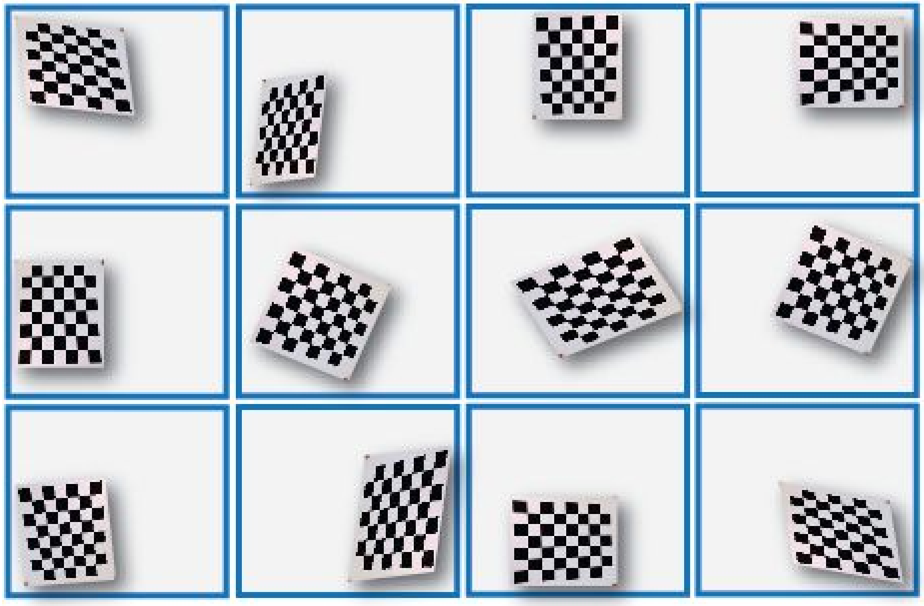
2) Note dimensions of target, size of square

3) Take multiple images in both cameras at same time by moving target to different locations on the image

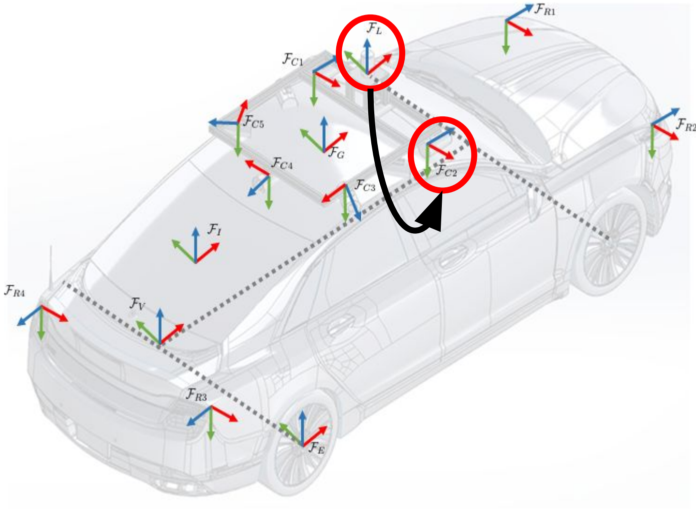
4) Calibrate using OpenCV, Matlab

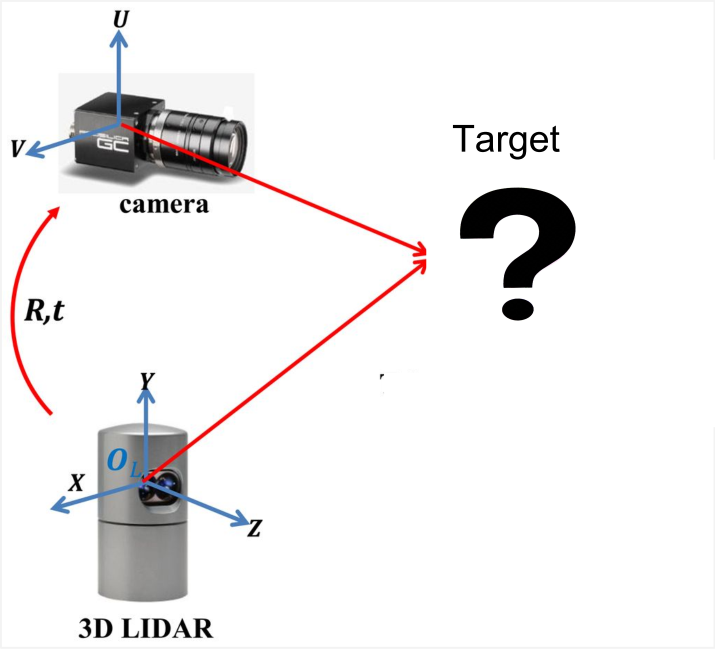
5) Reprojection error below 0.4 is good



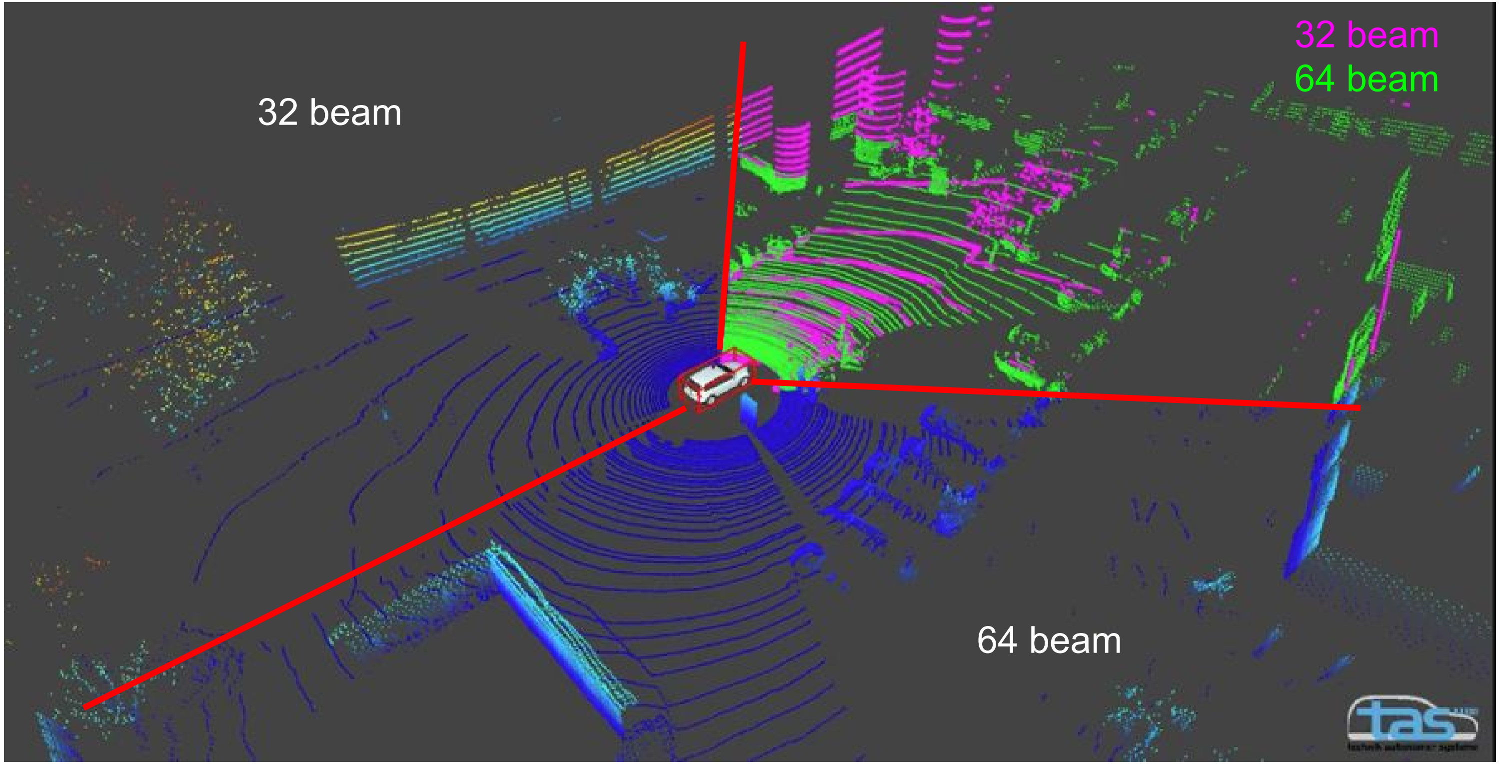



Questions?

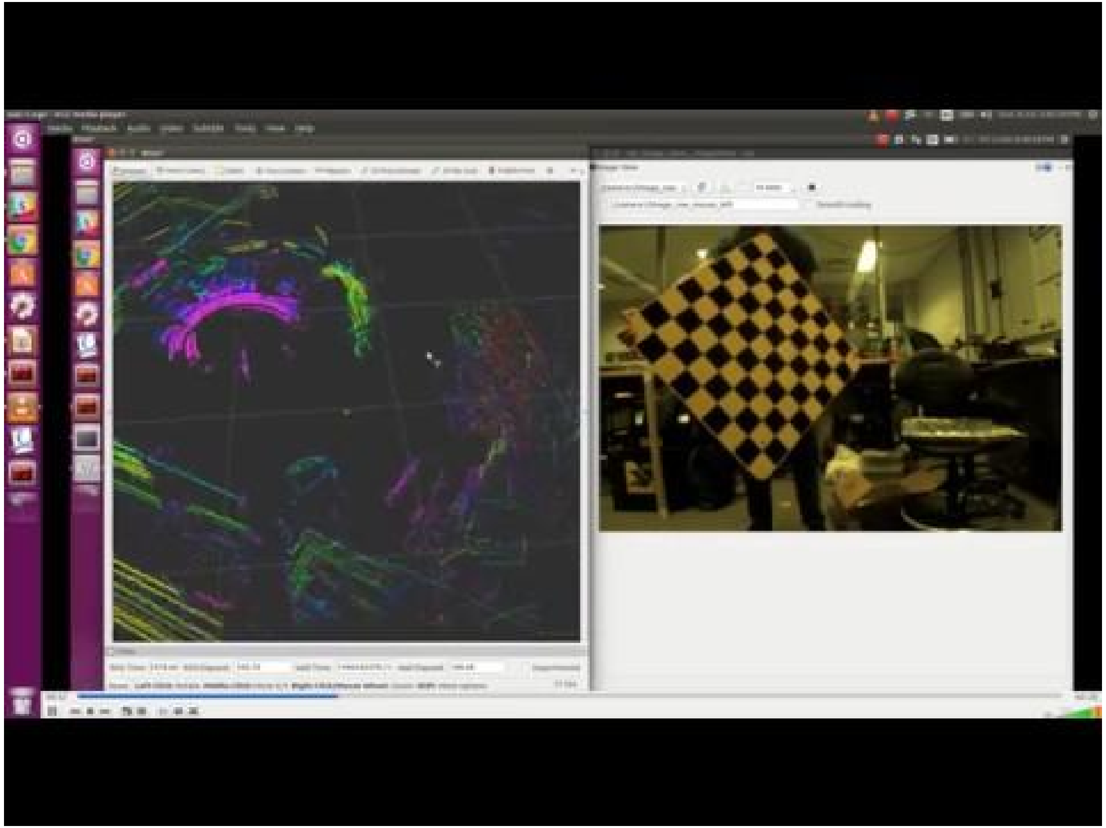


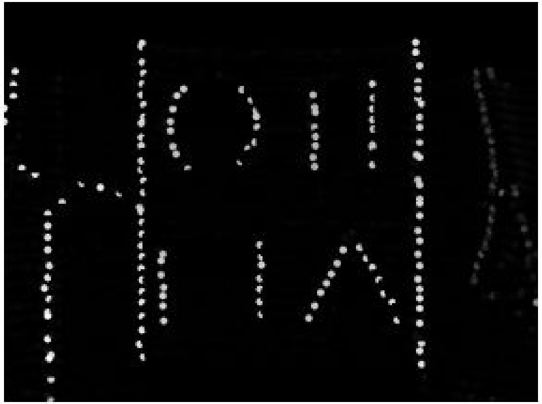


Where does this calibration differ from the extrinsic camera calibration ?
What sort of target should be used ?


32 beam
64 beam
32 beam
64 beam

Lidar response problems |





1) Given 3D points in world frame P^w and corresponding 2D pixel locations z_c in the camera

2) Estimate Transformations from world to the camera using PnP algorithm $T^{c:w}$.

3) Input initial transformation from lidar to camera (approximate). $T^{c:l}$

4) Transform 3D points from Lidar F_l to camera F_w via Extrinsic transformation.
$P^w = inv(\ T^{c:w}\ )T^{c:l}\ P^l$

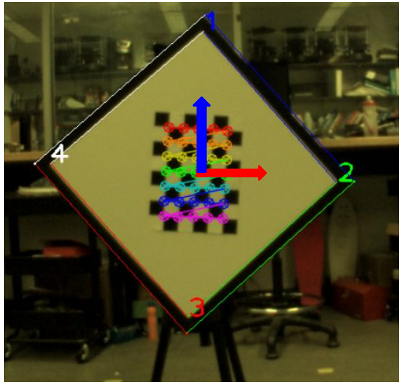
5) Determine which points $\mathbf{P^l}$ lie on target within some threshold.

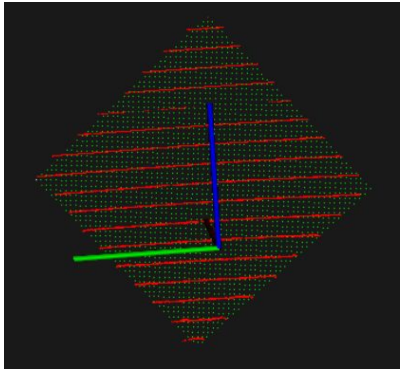
6) Project points on plane

7) Fit 'generated' point cloud to detected point cloud and get end-points.

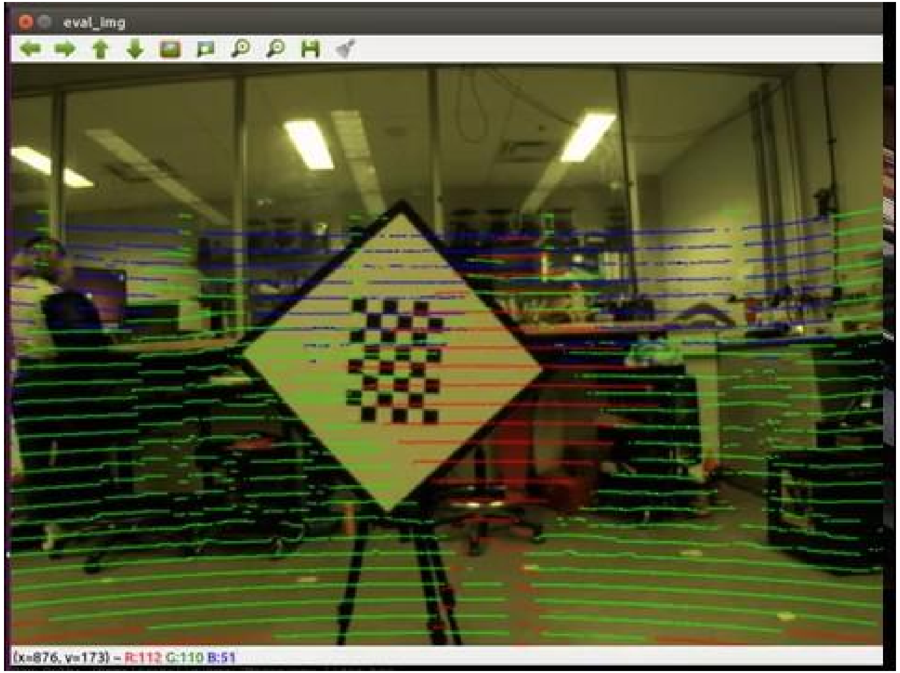
8) Determine end points of target on image

Reprojection Error: $\sum_{images} \sum_{points} d(\ \pi(T^{c:l}(\mathbf{P^l}\ ))\ ,\ z_c\ )^2$

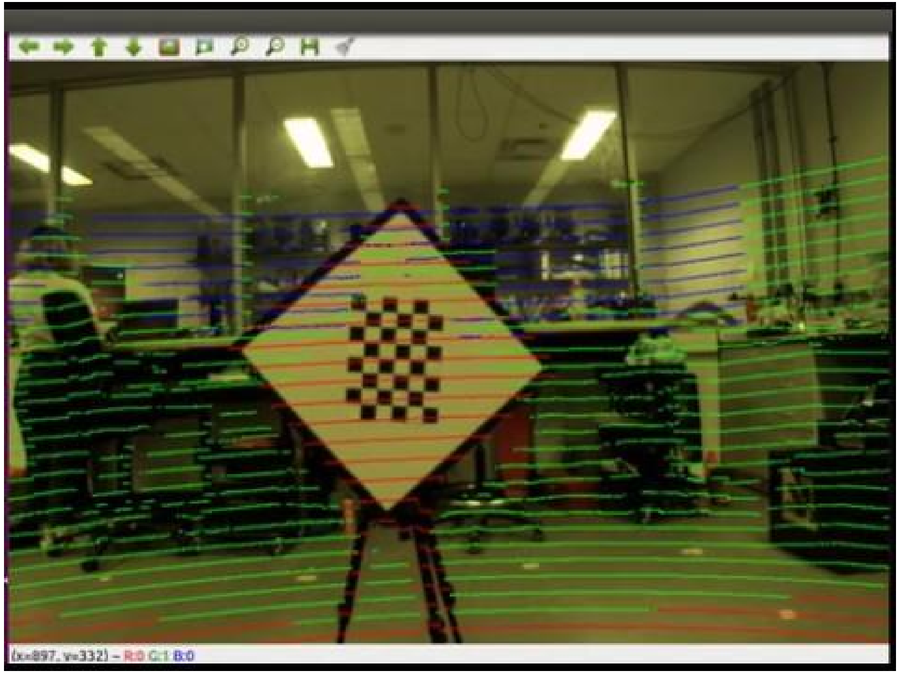




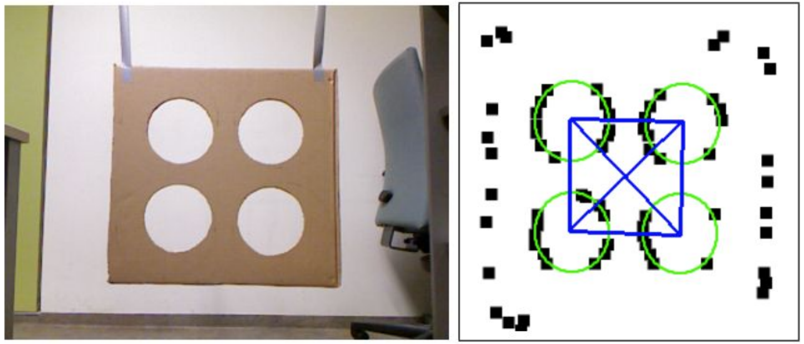
Before Calibration



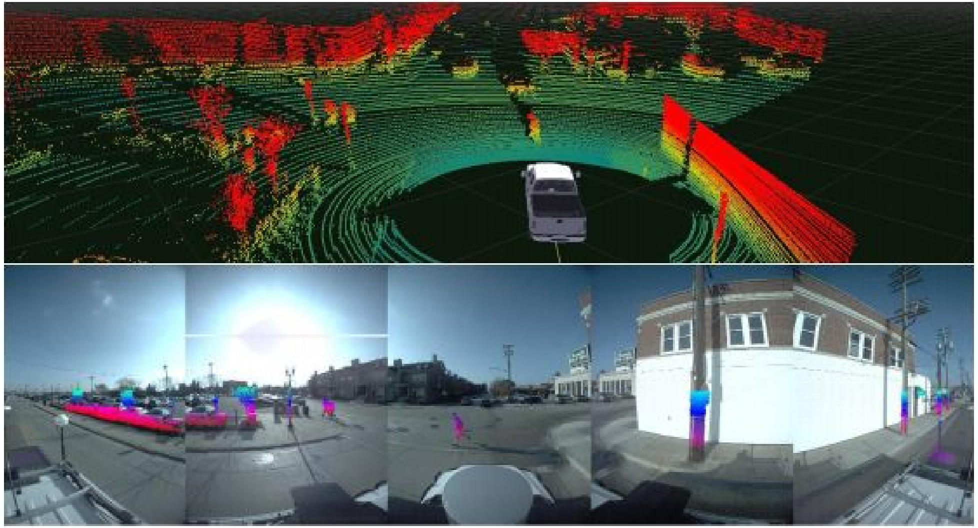
After Calibration



Alternative methods |



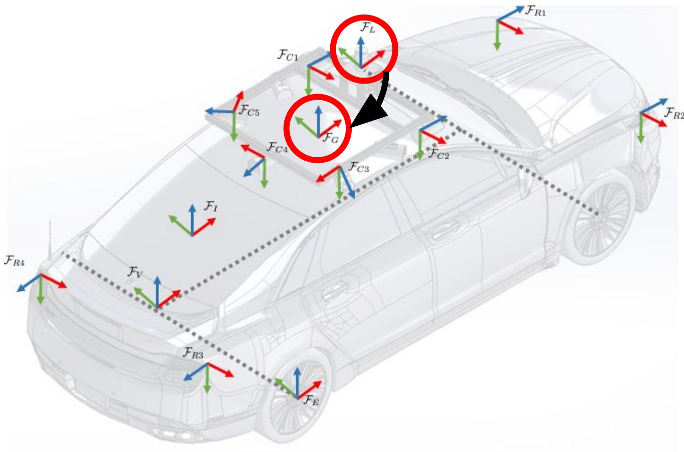
a) Martin Velas et.al Calibration of RGB Camera With Velodyne LiDAR

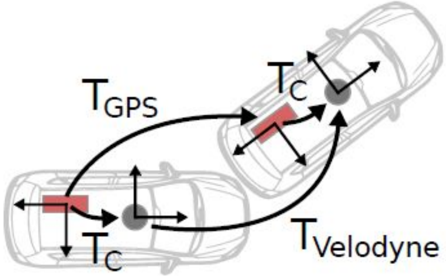


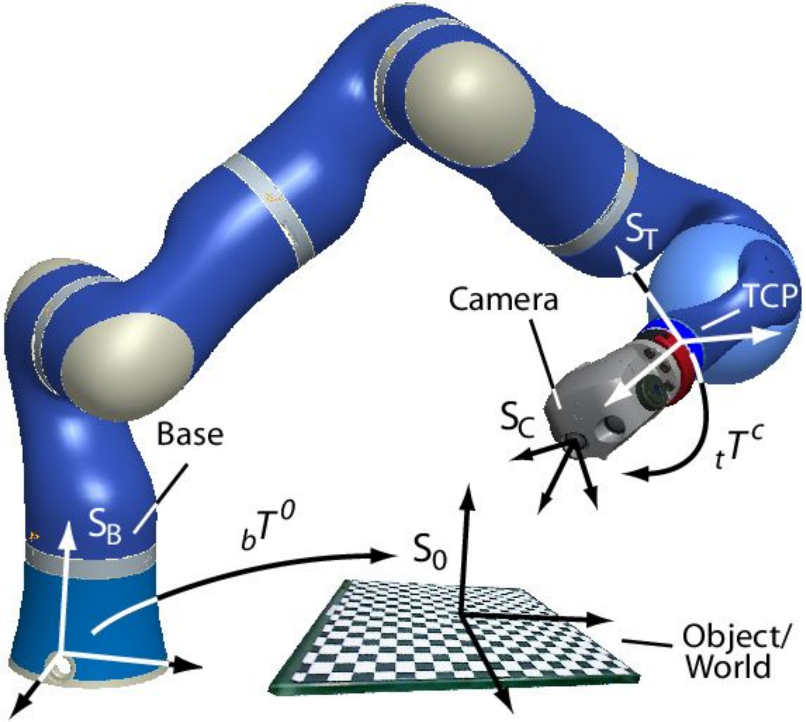
b) Gaurav Pandey et.al Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information

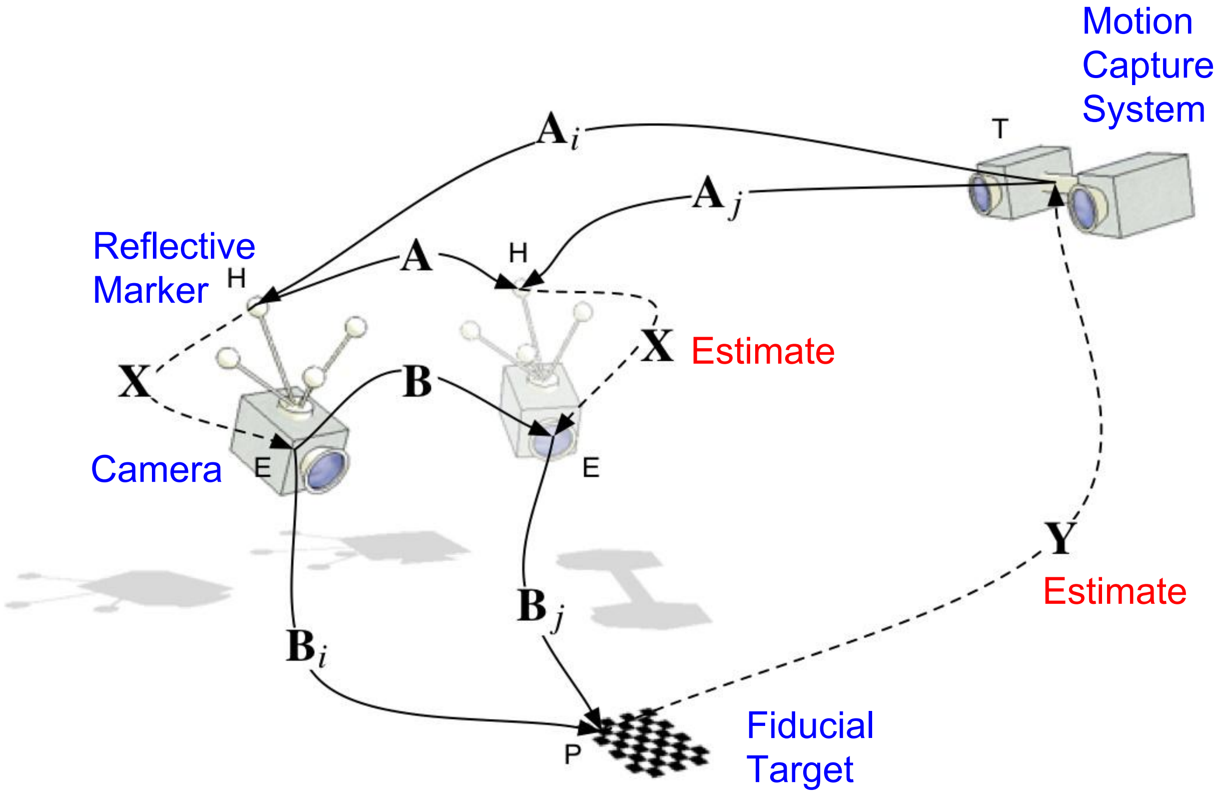
c) Jesse Levinson et.al Automatic Online Calibration of Cameras and Lasers

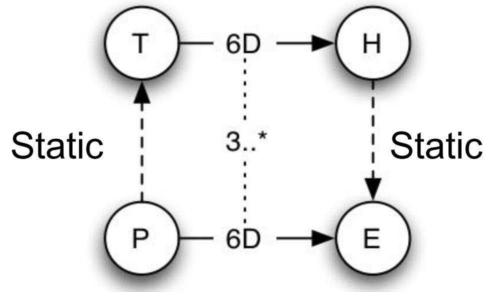
Questions |


$\mathcal{F}_{R1}$
$\mathcal{F}_{L}$
$\mathcal{F}_{C1}$
$\mathcal{F}_{C5}$
$\mathcal{F}_{G}$
$\mathcal{F}_{R2}$
$\mathcal{F}_{C4}$
$\mathcal{F}_{C3}$
$\mathcal{F}_{C2}$
$\mathcal{F}_{I}$
$\mathcal{F}_{R4}$
$\mathcal{F}_{V}$
$\mathcal{F}_{R3}$
$\mathcal{F}_{E}$


T_{GPS}
T_C
T_C
$T_{Velodyne}$


S_T
Camera
TCP
Base
S_C
${}_tT^c$
S_B
${}_bT^0$
S_0
Object/
World


Motion
Capture
System
T
A_i
A_j
Reflective
Marker
H
A
H
X
X
Estimate
B
Camera
E
E
B_j
B_i
Y
Estimate
Fiducial
Target
P


T
6D
H
Static
3..*
Static
P
6D
E

- T_{VM} : Marker to Vicon Motion Tracking

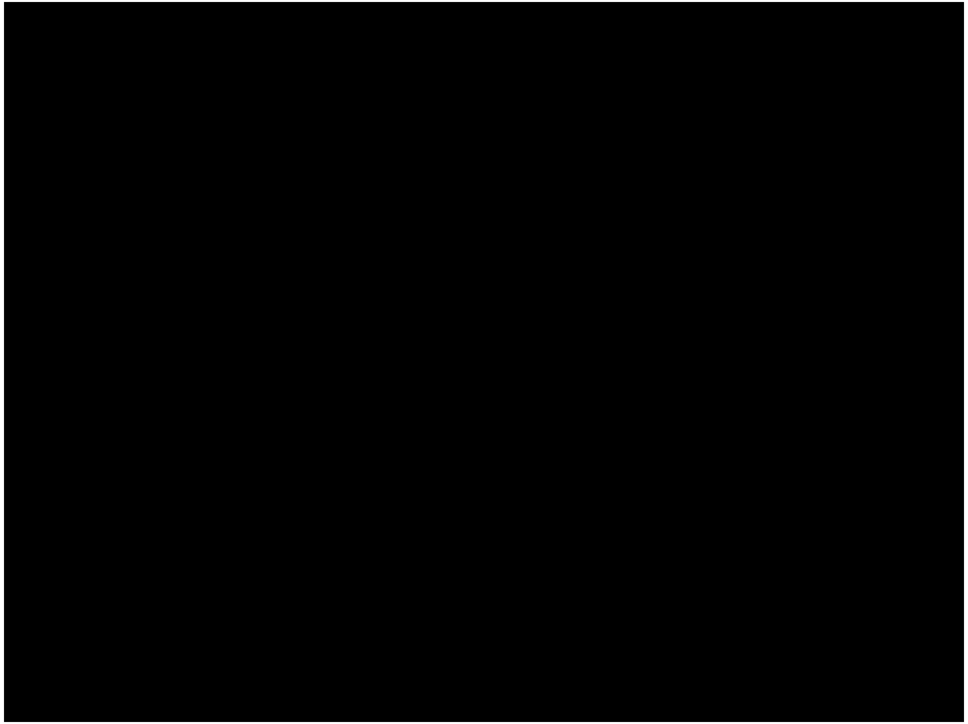
- T_{MC} : Camera to Marker

- T_{VF} : Fiducial Target to Vicon

- T_{FC} : Camera to Fiducial Target

$$C = \sum_{i=1}^{N} \Lambda(i)$$

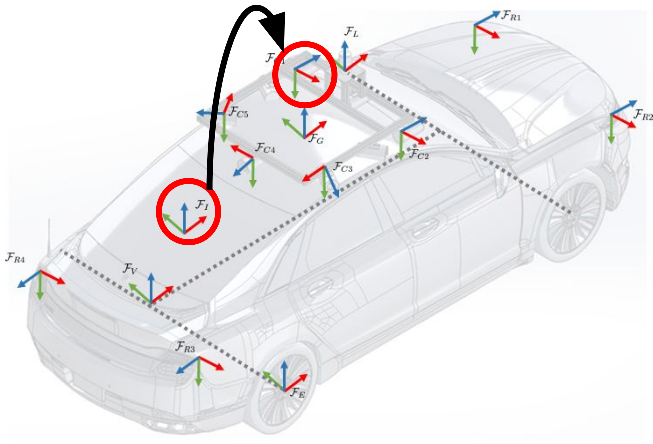
$$\Lambda(i) = (T_{VM}\hat{T}_{MC}) \boxminus (\hat{T}_{VF}T_{FC})$$

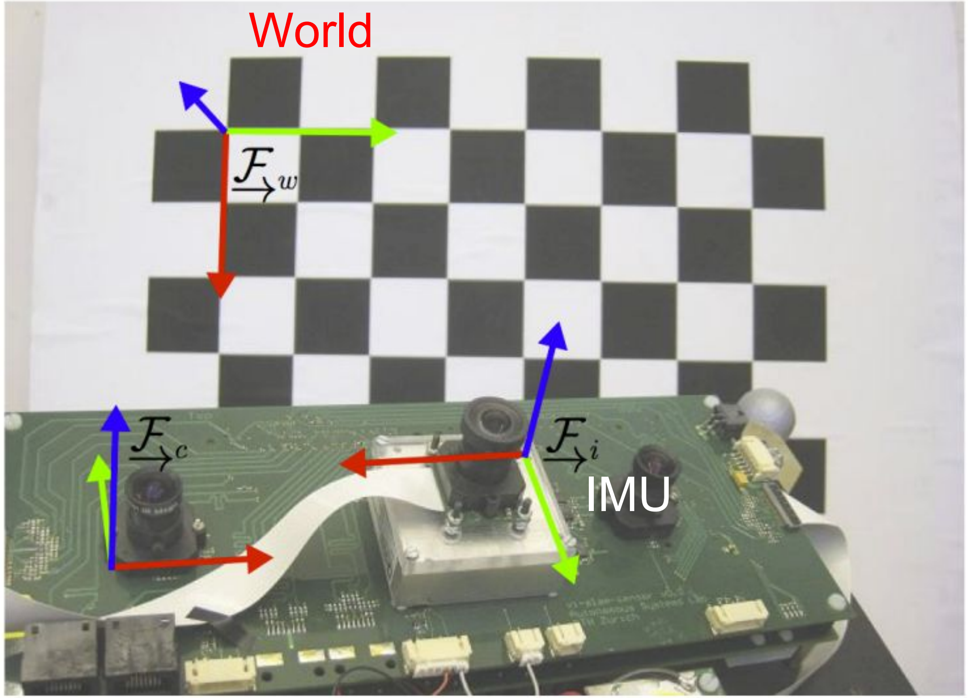


- Calibration with motion capture system and fiducial target similar to Lidar (SLAM) and GPS

- Planar motion leads to observability issues (Z, roll pitch)

- Need to find area that allows for sufficient excitation (car motion).



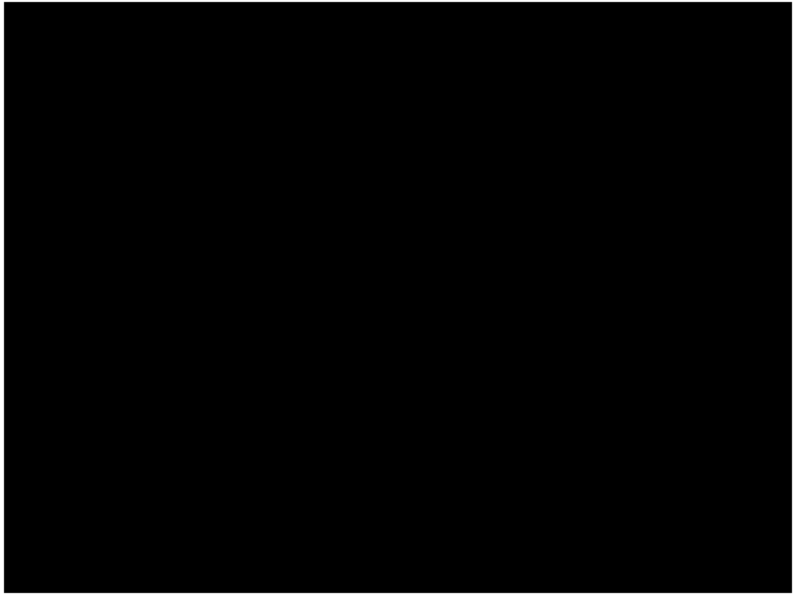

World
Camera
IMU

Quantities Estimated:

- Gravity direction expressed in World Frame
- Transformation between Camera and IMU
- Offset between Camera time and IMU time
- Pose of IMU
- Accelerometer and gyroscope biases

Assumptions Made:

- Camera Intrinsics are known
- IMU noise and bias models are known
- Have a guess for gravity in world frame
- Have a guess for the calibration matrix
- Geometry of calibration pattern is known
- Data association between image point and world point known



- Time varying states are represented as weighted sum of a finite number of known basis functions. phi(t) assumed to be known
- y_j is a measurement that arrived with timestamp t_j. h(.) is a measurement model that produces a predicted measurement from x(.) and d is unknown time offset
- The analytical Jacobian of the error term, needed for nonlinear least squares is derived by linearizing about a nominal value d, with respect to small changes in d.

$$\boldsymbol{\Phi}(t) := \begin{bmatrix} \phi_1(t) & \dots & \phi_B(t) \end{bmatrix}, \quad \mathbf{x}(t) := \boldsymbol{\Phi}(t)\mathbf{c},$$

$$\mathbf{e}_j := \mathbf{y}_j - \mathbf{h}\big(\mathbf{x}(t_j + d)\big),$$

$$\mathbf{e}_j \approx \mathbf{y}_j - \mathbf{h}\left(\boldsymbol{\Phi}(t_j + \bar{d})\mathbf{c}\right) - \mathbf{H}\dot{\boldsymbol{\Phi}}(t_j + \bar{d})\mathbf{c}\Delta d,$$

Accelerometer measurement, gyroscope measurement (at time k) pixel location (time t+d), h(.) projection model, n is noise. J is number of images

$$\boldsymbol{\alpha}_k := \mathbf{C}\left(\boldsymbol{\varphi}(t_k)\right)^T \left(\mathbf{a}(t_k) - \mathbf{g}_w\right) + \mathbf{b}_a(t_k) + \mathbf{n}_{a_k},$$
$$\boldsymbol{\varpi}_k := \mathbf{C}\left(\boldsymbol{\varphi}(t_k)\right)^T \boldsymbol{\omega}(t_k) + \mathbf{b}_\omega(t_k) + \mathbf{n}_{\omega_k},$$
$$\mathbf{y}_{mj} := \mathbf{h}\left(\mathbf{T}_{c,i}\mathbf{T}_{w,i}(t_j + d)^{-1}\mathbf{p}_w^m\right) + \mathbf{n}_{y_{mj}},$$

IMU biases

$$\dot{\mathbf{b}}_a(t) = \mathbf{w}_a(t) \qquad \mathbf{w}_a(t) \sim \mathcal{GP}\left(\mathbf{0}, \mathbf{Q}_a\delta(t - t')\right)$$
$$\dot{\mathbf{b}}_\omega(t) = \mathbf{w}_\omega(t) \qquad \mathbf{w}_\omega(t) \sim \mathcal{GP}\left(\mathbf{0}, \mathbf{Q}_\omega\delta(t - t')\right)$$

Error terms are associated with the measurements constructed as the difference between the measurement and the predicted measurement for current state estimate

$$\mathbf{e}_{y_{mj}} := \mathbf{y}_{mj} - \mathbf{h}\left(\mathbf{T}_{c,i}\mathbf{T}_{w,i}(t_j + d)^{-1}\mathbf{p}_w^m\right)$$
$$J_y := \frac{1}{2}\sum_{j=1}^{J}\sum_{m=1}^{M} \mathbf{e}_{y_{mj}}^T \mathbf{R}_{y_{mj}}^{-1} \mathbf{e}_{y_{mj}}$$
$$\mathbf{e}_{\alpha_k} := \boldsymbol{\alpha}_k - \mathbf{C}\left(\boldsymbol{\varphi}(t_k)\right)^T \left(\mathbf{a}(t_k) - \mathbf{g}_w\right) + \mathbf{b}_a(t_k)$$
$$J_\alpha := \frac{1}{2}\sum_{k=1}^{K} \mathbf{e}_{\alpha_k}^T \mathbf{R}_{\alpha_k}^{-1} \mathbf{e}_{\alpha_k}$$
$$\mathbf{e}_{\omega_k} := \boldsymbol{\varpi}_k - \mathbf{C}\left(\boldsymbol{\varphi}(t_k)\right)^T \boldsymbol{\omega}(t_k) + \mathbf{b}_\omega(t_k)$$
$$J_\omega := \frac{1}{2}\sum_{k=1}^{K} \mathbf{e}_{\omega_k}^T \mathbf{R}_{\omega_k}^{-1} \mathbf{e}_{\omega_k}$$
$$\mathbf{e}_{b_a}(t) := \dot{\mathbf{b}}_a(t)$$
$$J_{b_a} := \frac{1}{2}\int_{t_1}^{t_K} \mathbf{e}_{b_a}(\tau)^T \mathbf{Q}_a^{-1} \mathbf{e}_{b_a}(\tau)\, d\tau$$
$$\mathbf{e}_{b_\omega}(t) := \dot{\mathbf{b}}_\omega(t)$$
$$J_{b_\omega} := \frac{1}{2}\int_{t_1}^{t_K} \mathbf{e}_{b_\omega}(\tau)^T \mathbf{Q}_\omega^{-1} \mathbf{e}_{b_\omega}(\tau)\, d\tau$$