Forward Kinematics: Product of Exponentials

Lecture 5

Winter 2023

M | ROBOTICS

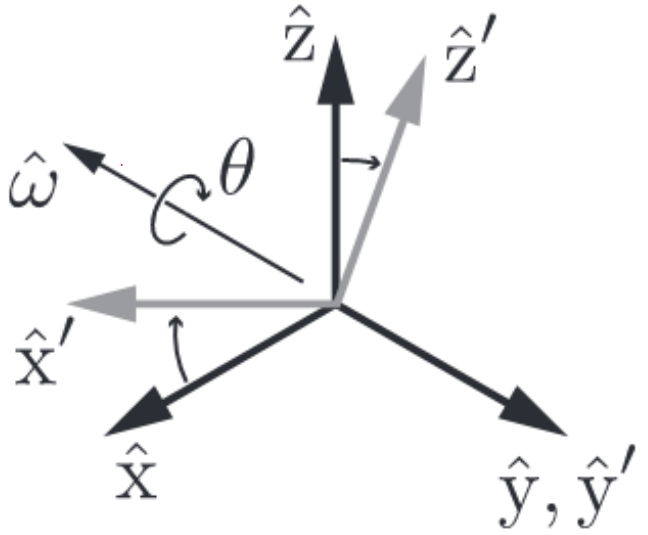
3D Rotation

Rotation around any single axis:

$$R_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$



Rotation around arbitrary axis:

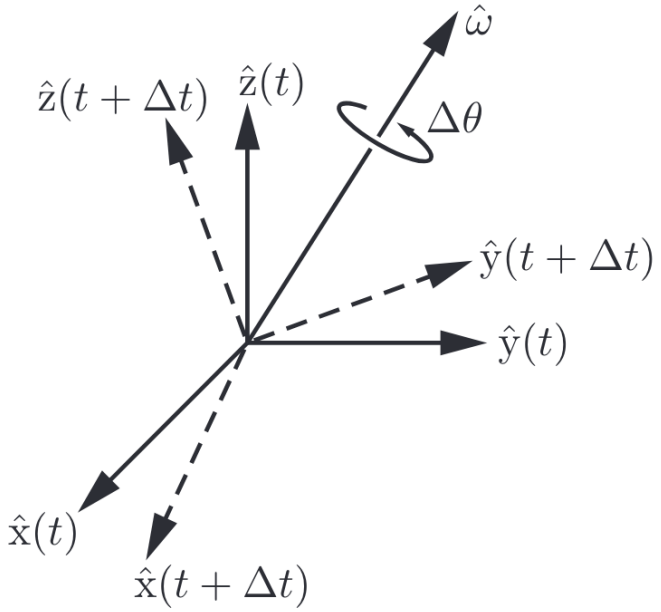
$$\hat{\omega} = (\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3)$$

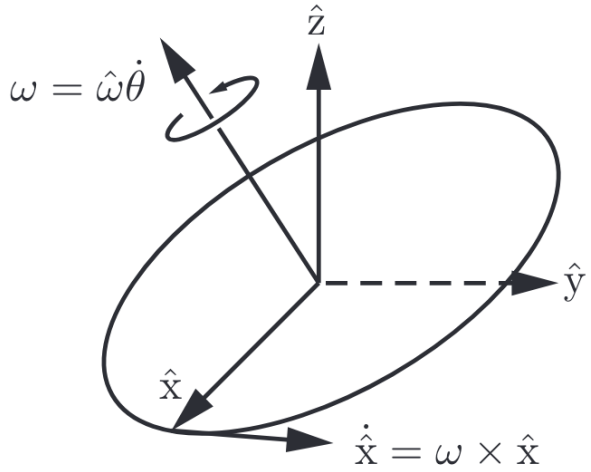
$$\mathrm{Rot}(\hat{\omega}, \theta) =$$

$$\begin{bmatrix} \mathrm{c}_\theta + \hat{\omega}_1^2(1-\mathrm{c}_\theta) & \hat{\omega}_1\hat{\omega}_2(1-\mathrm{c}_\theta) - \hat{\omega}_3\mathrm{s}_\theta & \hat{\omega}_1\hat{\omega}_3(1-\mathrm{c}_\theta) + \hat{\omega}_2\mathrm{s}_\theta \\ \hat{\omega}_1\hat{\omega}_2(1-\mathrm{c}_\theta) + \hat{\omega}_3\mathrm{s}_\theta & \mathrm{c}_\theta + \hat{\omega}_2^2(1-\mathrm{c}_\theta) & \hat{\omega}_2\hat{\omega}_3(1-\mathrm{c}_\theta) - \hat{\omega}_1\mathrm{s}_\theta \\ \hat{\omega}_1\hat{\omega}_3(1-\mathrm{c}_\theta) - \hat{\omega}_2\mathrm{s}_\theta & \hat{\omega}_2\hat{\omega}_3(1-\mathrm{c}_\theta) + \hat{\omega}_1\mathrm{s}_\theta & \mathrm{c}_\theta + \hat{\omega}_3^2(1-\mathrm{c}_\theta) \end{bmatrix}$$

Angular Velocity

Consider instantaneous angular velocity about an axis





Angular velocity:

$$\omega = \hat{\omega}\dot{\theta}$$

$$\dot{\hat{x}} = \omega \times \hat{x}$$

$$\dot{\hat{y}} = \omega \times \hat{y}$$

$$\dot{\hat{z}} = \omega \times \hat{z}$$

Angular Velocity – Rotation Matrix

- Must choose reference frame to describe ω
- Let $\{s\}$ be the fixed "world" or "space" frame and $\{b\}$ be the "body" frame
- Let $R(t)$ be the rotation matrix describing orientation of the body frame $\{b\}$ in the space frame $\{s\}$

$$R(t) = \begin{bmatrix} r_1(t) & r_2(t) & r_3(t) \end{bmatrix}$$

- Where $r_1(t), r_2(t), r_3(t)$ describe $\hat{x}, \hat{y}, \hat{z}$ in fixed-frame coordinates

Angular Velocity – Rotation Matrix

- Given the rotation R at time t:

$$R(t) = \begin{bmatrix} r_1(t) & r_2(t) & r_3(t) \end{bmatrix}$$

- The time rate of change of R is:

$$\dot{R} = [\omega_s \times r_1 \;\; \omega_s \times r_2 \;\; \omega_s \times r_3] = \omega_s \times R$$

Skew Symmetric Matrix

- Instead of using cross product…

$$\dot{R} = [\omega_s \times r_1 \;\; \omega_s \times r_2 \;\; \omega_s \times r_3] = \omega_s \times R$$

- We can define a **skew symmetric matrix** to produce the same result

$$\text{Given a vector } x = [x_1 \; x_2 \; x_3]^{\mathrm{T}} \in \mathbb{R}^3, \text{ define}$$

$$[x] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}.$$

Matrix Exponential

Now consider the vector linear differential equation

$$\dot{x}(t) = Ax(t), \tag{3.40}$$

where $x(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$ is constant, and the initial condition $x(0) = x_0$ is given. From the above scalar result one can conjecture a solution of the form

$$x(t) = e^{At}x_0 \tag{3.41}$$

where the **matrix exponential** e^{At} now needs to be defined in a meaningful way. Again mimicking the scalar case, we define the matrix exponential to be

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \cdots \tag{3.42}$$

Exponential Representation of Rotation

- Consider the angular velocity of a point

$$\dot{p} = [\hat{\omega}]p$$

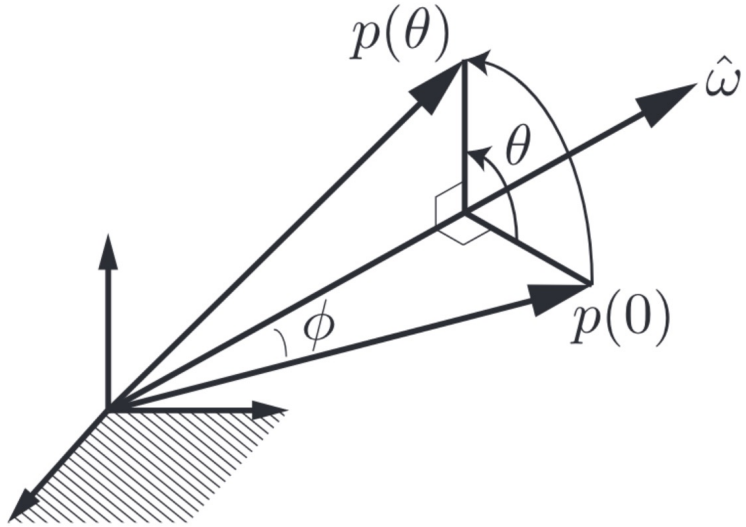
- A solution to this differential equation is

$$p(t) = e^{[\hat{\omega}]t}p(0)$$

- If we normalize velocity (e.g. 1 rad/s)

$$p(\theta) = e^{[\hat{\omega}]\theta}p(0)$$

- Interpret as rotation from $t = 0$ to $t = \theta$ at 1 rad/s



Rodrigues' Formula

$$
\begin{aligned}
e^{[\hat{\omega}]\theta} &= I + [\hat{\omega}]\theta + [\hat{\omega}]^2 \frac{\theta^2}{2!} + [\hat{\omega}]^3 \frac{\theta^3}{3!} + \cdots \\
&= I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots\right)[\hat{\omega}] + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \cdots\right)[\hat{\omega}]^2
\end{aligned}
$$

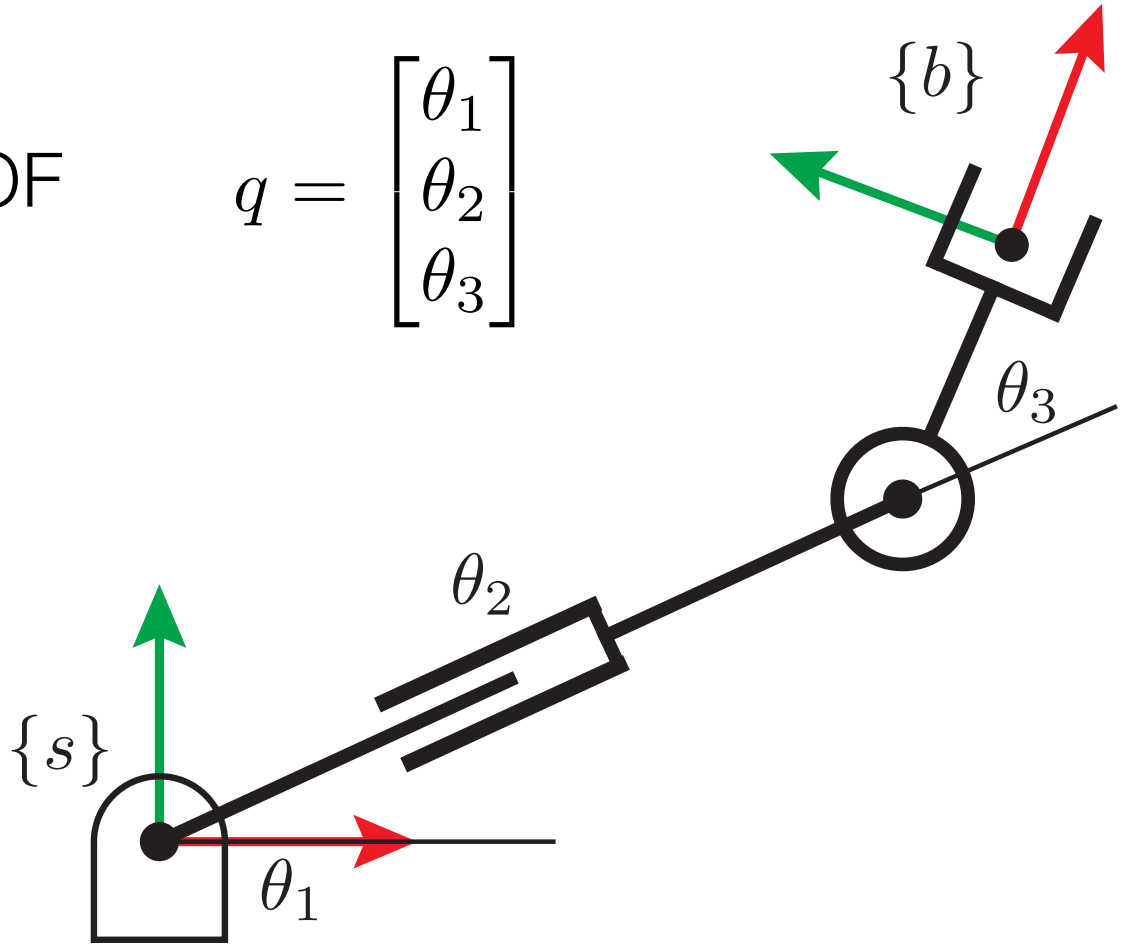
$$
\begin{aligned}
\sin\theta &= \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots \\
\cos\theta &= 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots
\end{aligned}
$$

$$
\boxed{\mathrm{Rot}(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta} = I + \sin\theta\,[\hat{\omega}] + (1 - \cos\theta)[\hat{\omega}]^2 \in SO(3)}
$$

Forward Kinematics with PoX

- Define home position
- Find screw vector for each DOF
- Use PoX formula to find pose

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

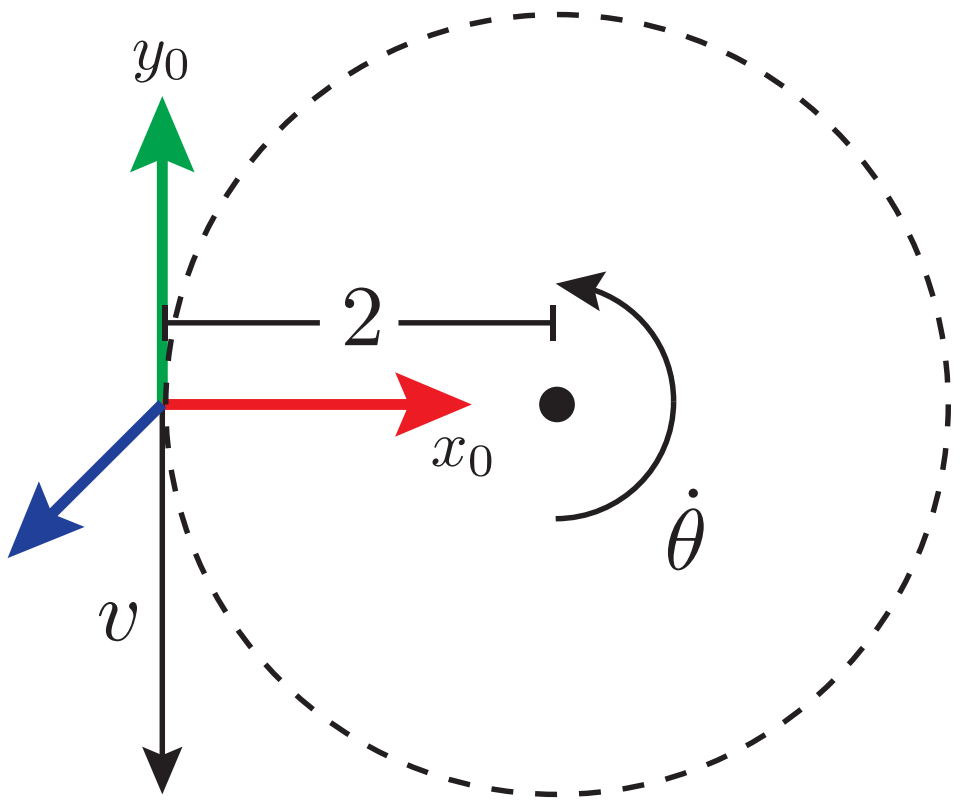


Screws

- ω is angular velocity about axis of rotation
- v is linear velocity of origin

$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \mathcal{S}\dot{\theta}$$

- Screw is normalized twist



$$\mathcal{S} = \begin{bmatrix} \mathcal{S}_\omega \\ \mathcal{S}_v \end{bmatrix} = \begin{bmatrix} \text{angular velocity when } \dot{\theta} = 1 \\ \text{linear velocity of origin when } \dot{\theta} = 1 \end{bmatrix}$$

Constructing Rigid Body Transformations

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in se(3) \qquad [\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in so(3)$$

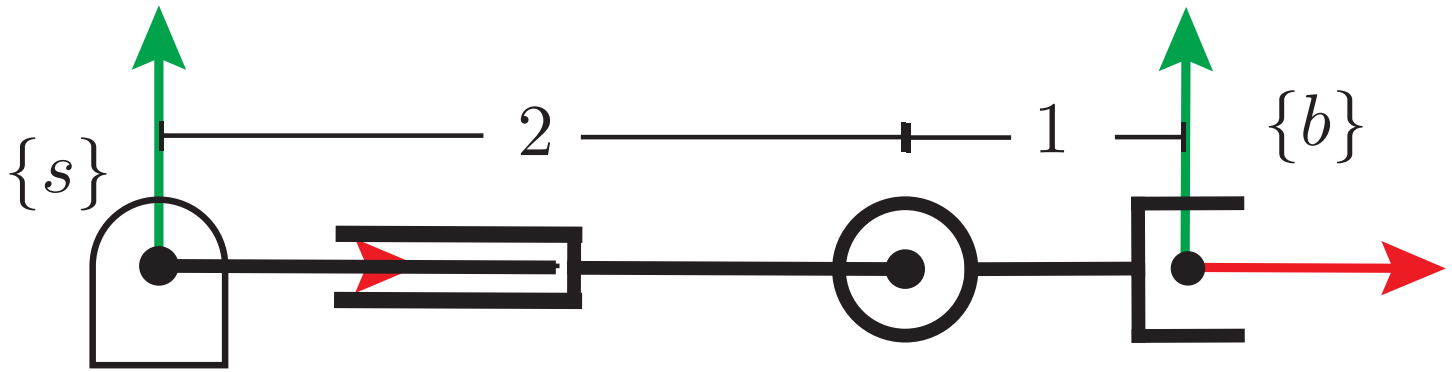
$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & \left(I\theta + (1-\cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2\right) v \\ 0 & 1 \end{bmatrix}$$

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

Home Position

- M is home position, where all joint variables are 0

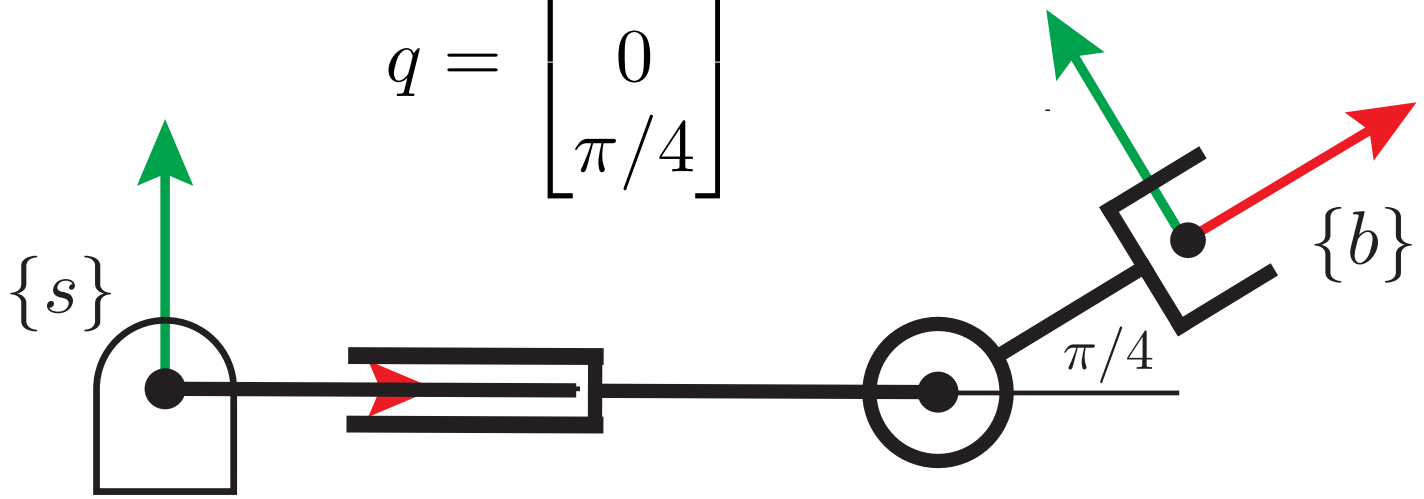
$$M = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Screw Vector – Joint 3

- How does the motion about the joint axis move a point on the $\{s\}$ frame?

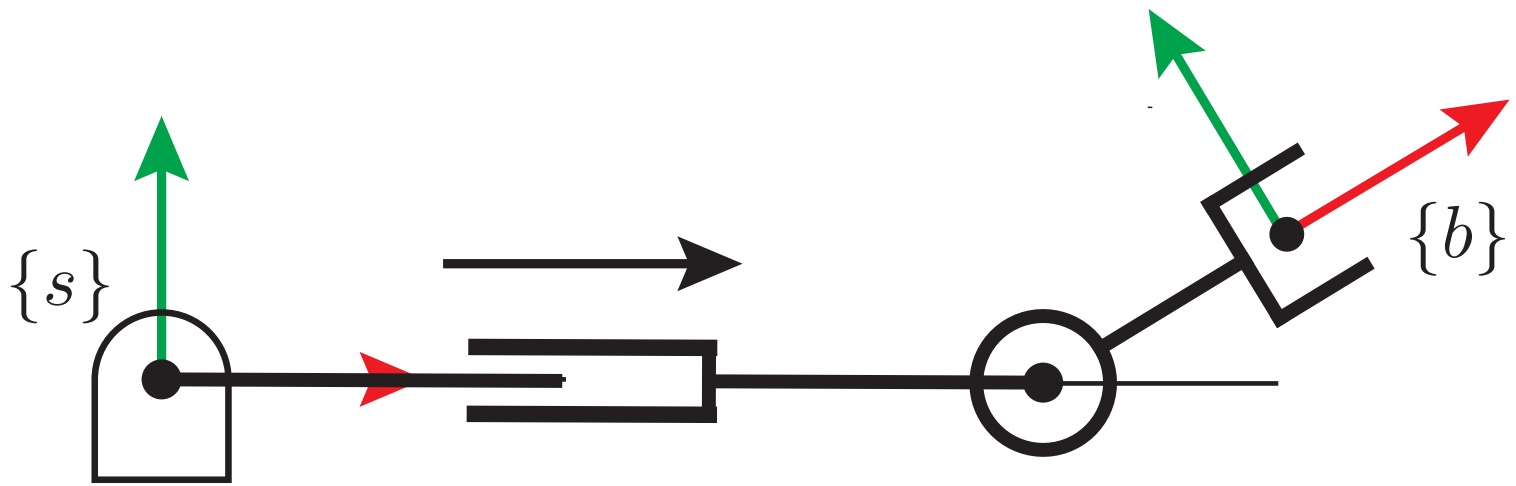
$$q = \begin{bmatrix} 0 \\ 0 \\ \pi/4 \end{bmatrix}$$



$$\mathcal{S}_3 = \begin{bmatrix} \omega_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -2 \\ 0 \end{bmatrix}$$

Screw Vector – Joint 2

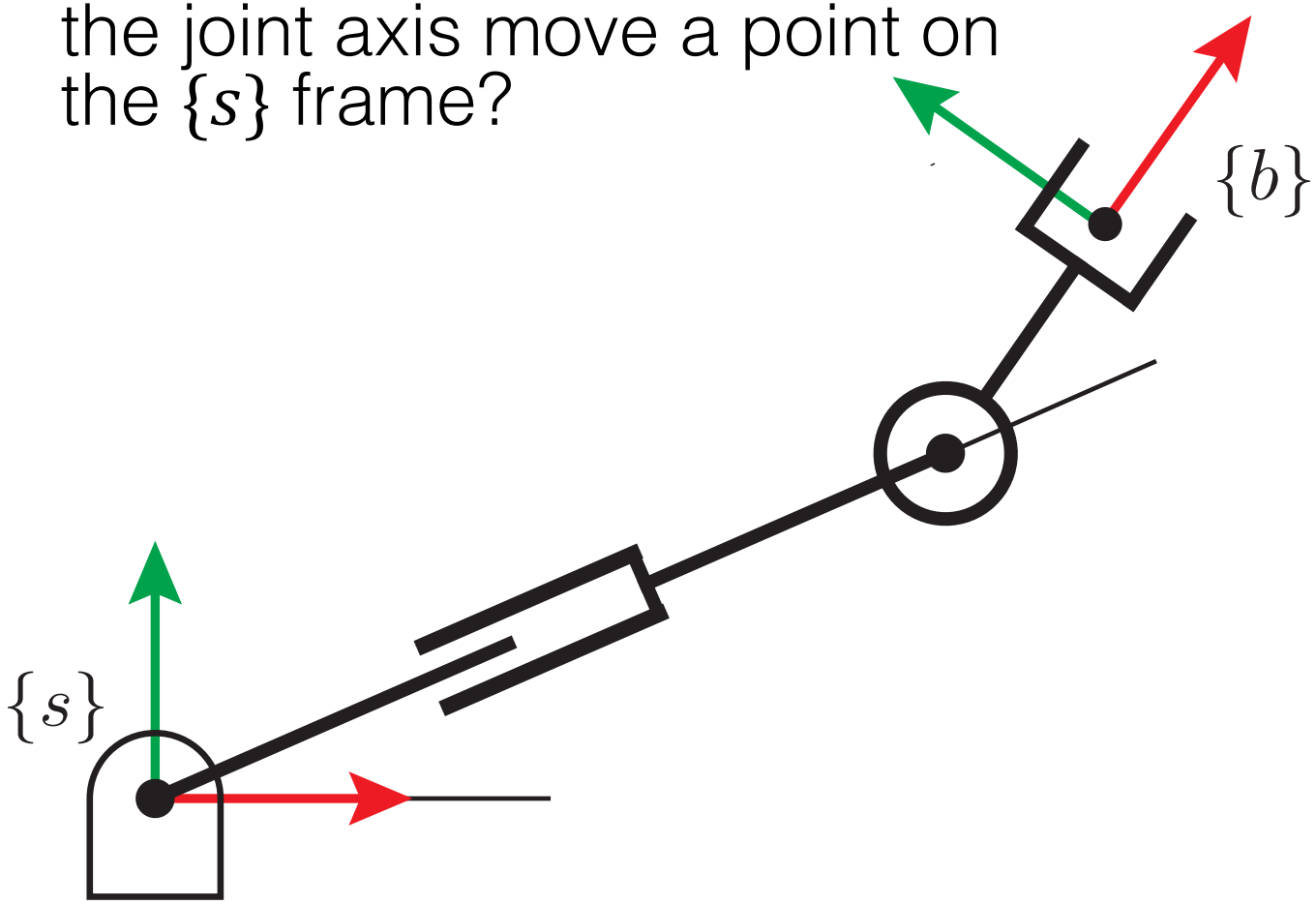
- How does the motion about the joint axis move a point on the $\{s\}$ frame?



$$\mathcal{S}_2 = \begin{bmatrix} \omega_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Screw Vector – Joint 1

- How does the motion about the joint axis move a point on the $\{s\}$ frame?



$$\mathcal{S}_1 = \begin{bmatrix} \omega_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Forward Kinematic Map

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in se(3) \qquad q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$T(q) = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M$$

Summary

- Forward kinematic map can be found from a product of matrix exponentials
- This requires no intermediate frames (like DH)

Workspace Reconstruction

Lecture 5.2

Winter 2023

M | ROBOTICS

Full Camera Matrix
Workspace to Camera

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix}$$

intrinsic

extrinsic

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & | & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

projection

Camera to Image Frame (Intrinsic)

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R} & & \mathbf{T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

World to Camera (Extrinsic)

Pixel to World frame

- With depth information we can undo the scale by $1/Z_c$

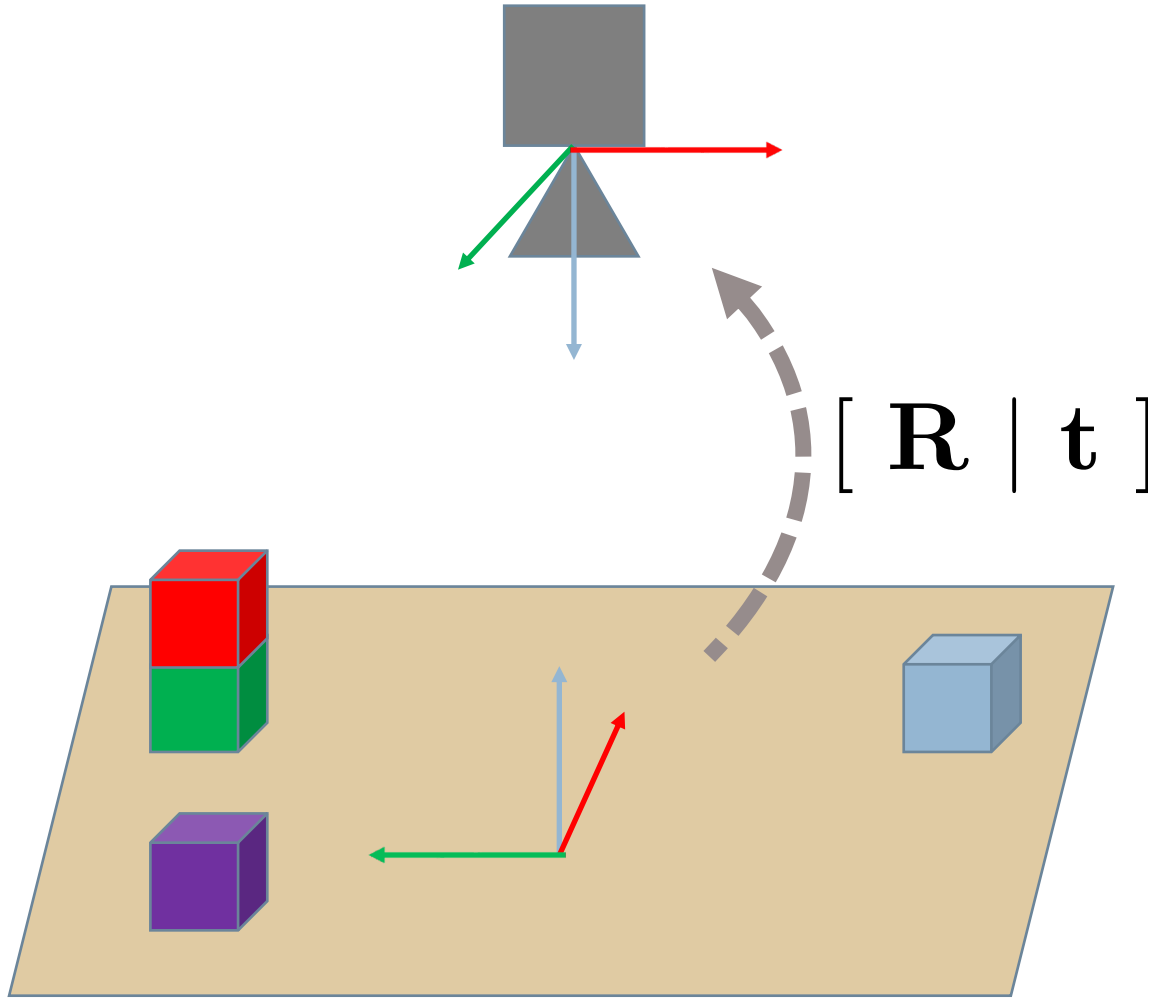
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = Z_c(u, v)\ \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Change to homogeneous coordinates and multiply inverse extrinsic matrix

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

Finding the Extrinsics

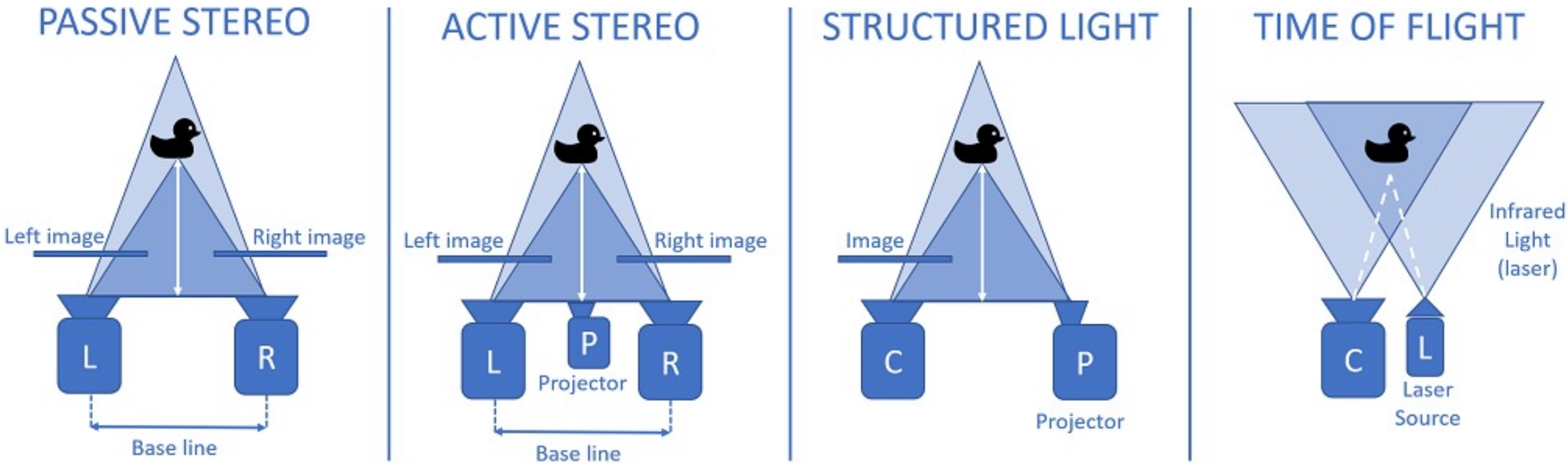
- Need camera extrinsic transformation (world to camera)
- Carefully position camera and measure (naïve)
- Calculate from pixel/point correspondances (cv2.solvePnP)
- Calculate from point/point correspondances (least squares, 3D affine)
- Use fiducials (Apriltags)



Getting depth information

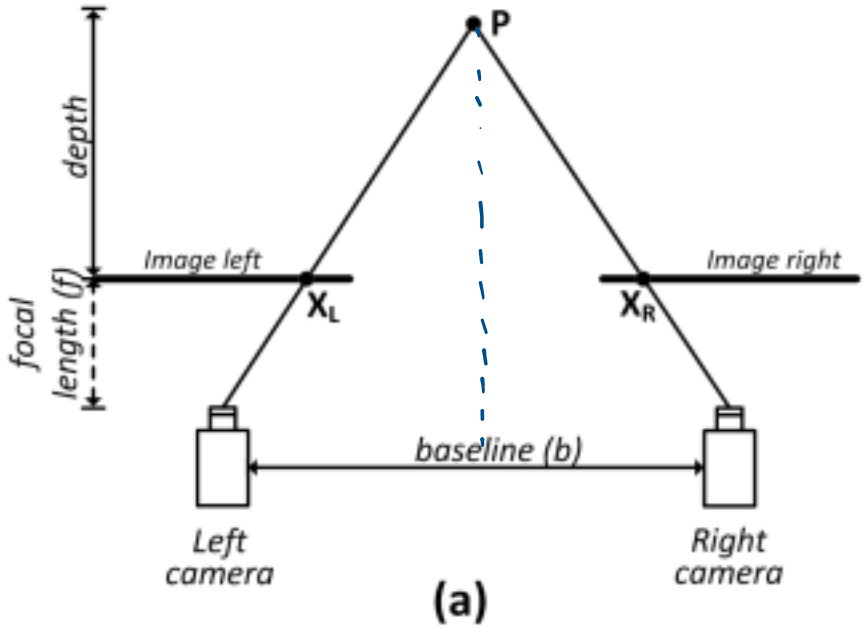
- Depth from scale (if objects viewed are of known size)
  - Fiducials / solvePnP
  - Sensitive to camera resolution
  - Less accurate as objects get further away
- Depth from stereo discrepancy
  - Stereo cameras
- Depth from structured light distortion
  - Kinect / Realsense
- Can get depth from direct measurement (Lidar / Radar)
  - Realsense Lidar camera

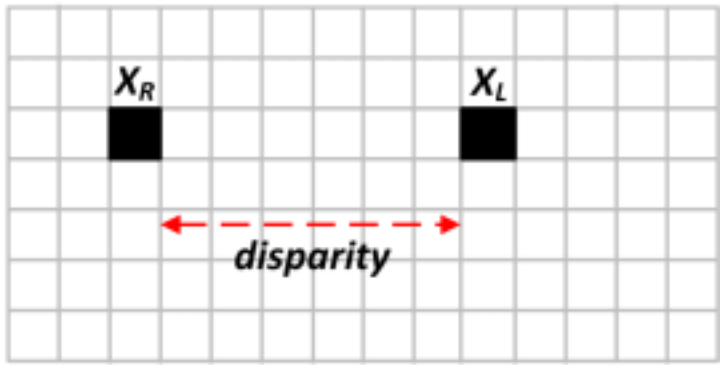
Types of depth cameras

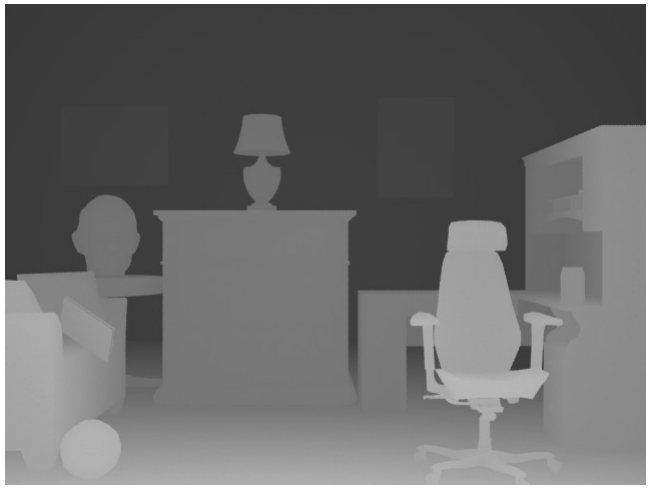


Stereo Camera

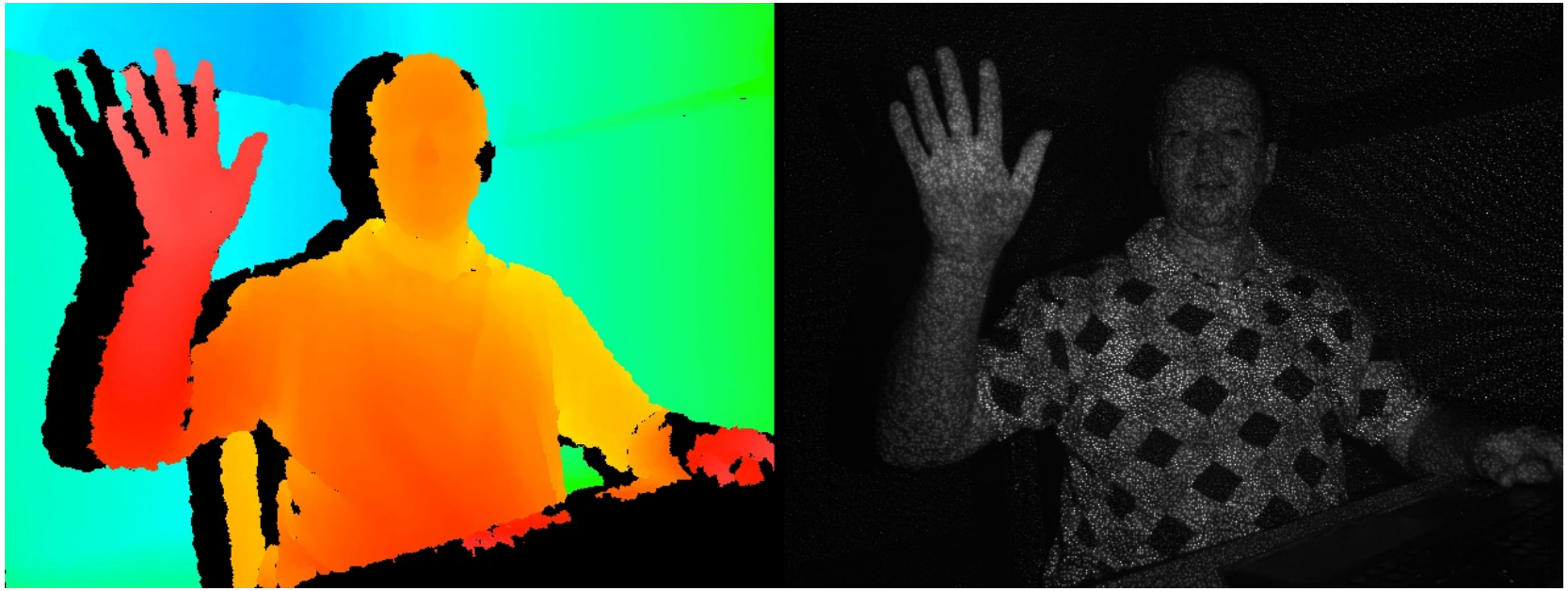
- Create a disparity map to get depth



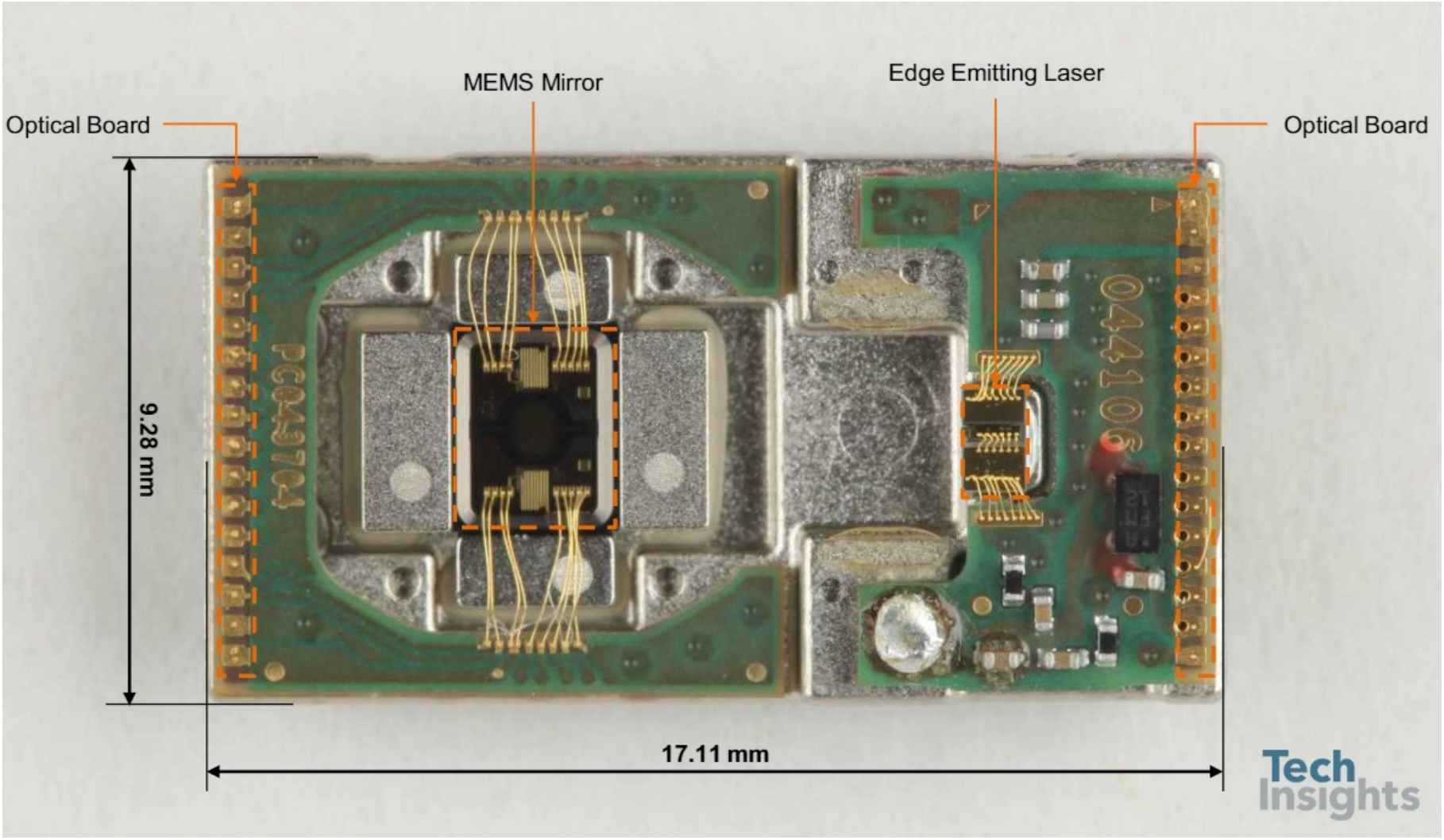




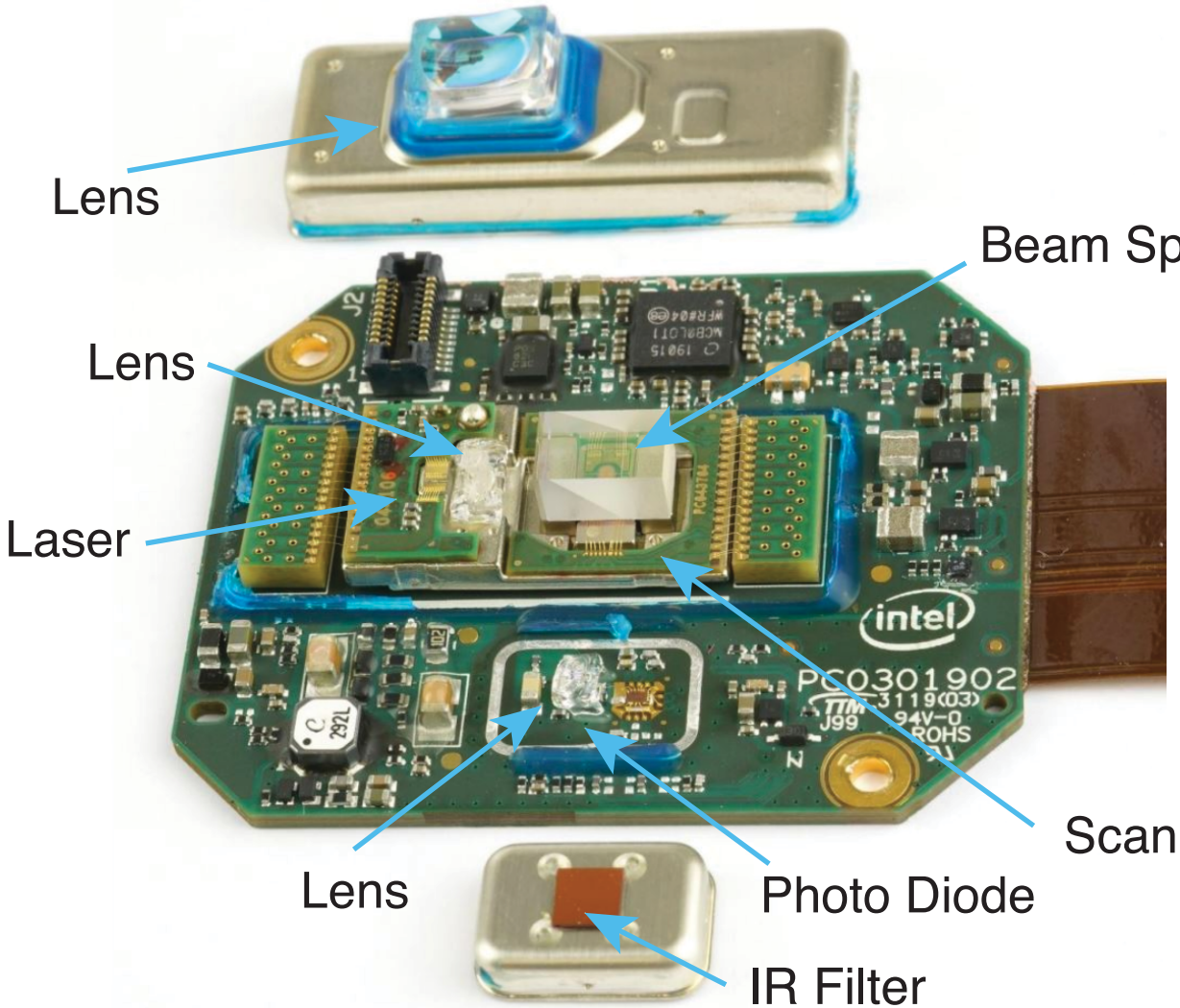
Structured Light Cameras

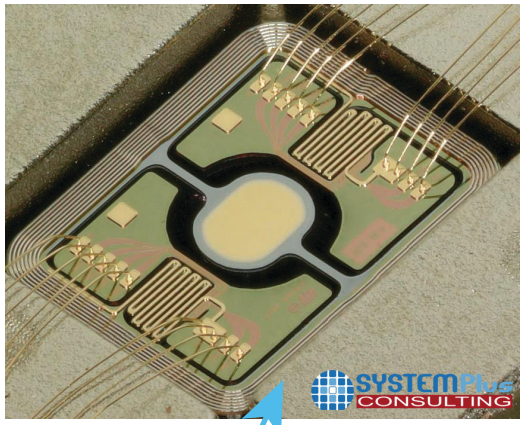


Realsense LIDAR



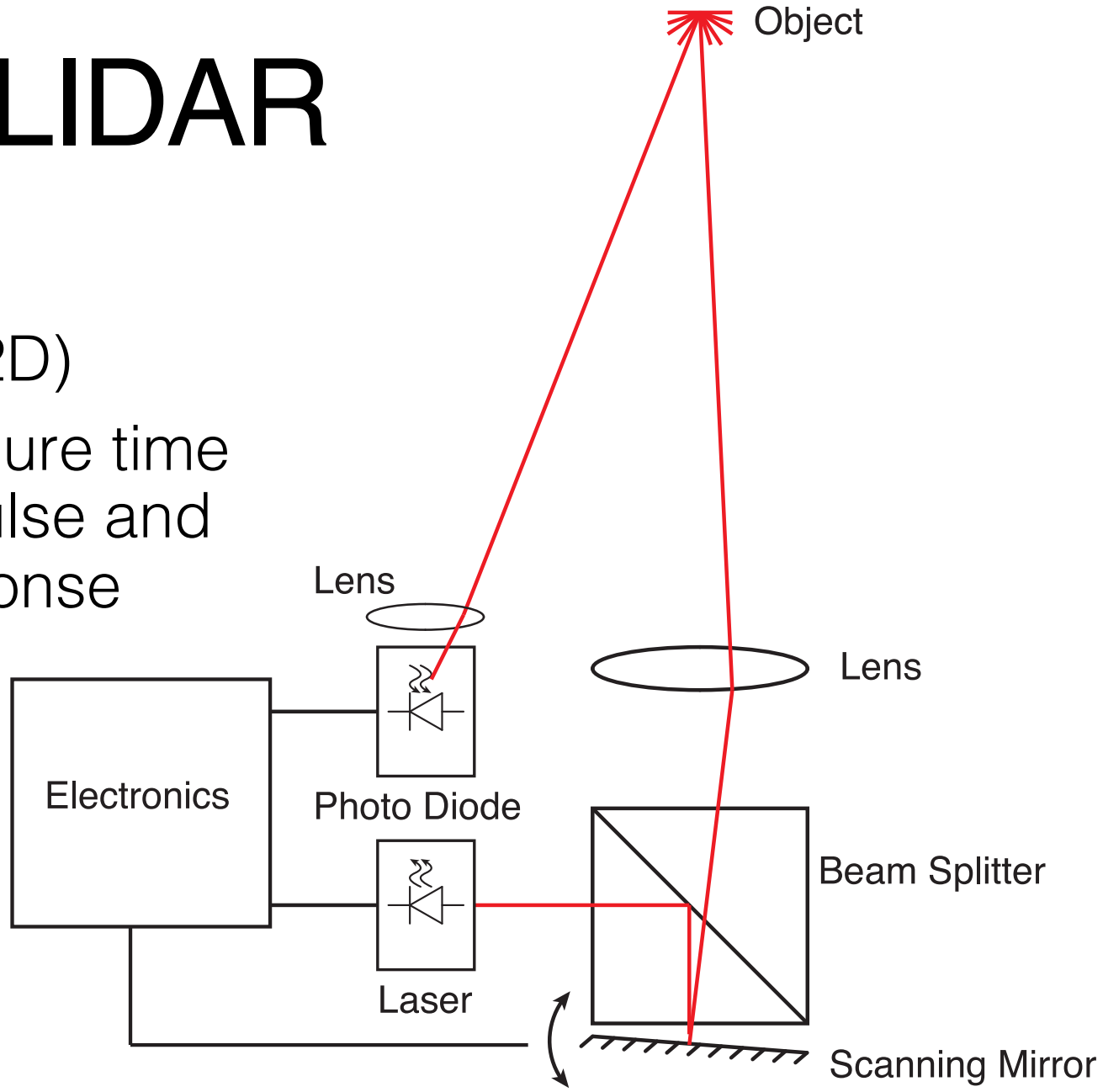
Inside the L515





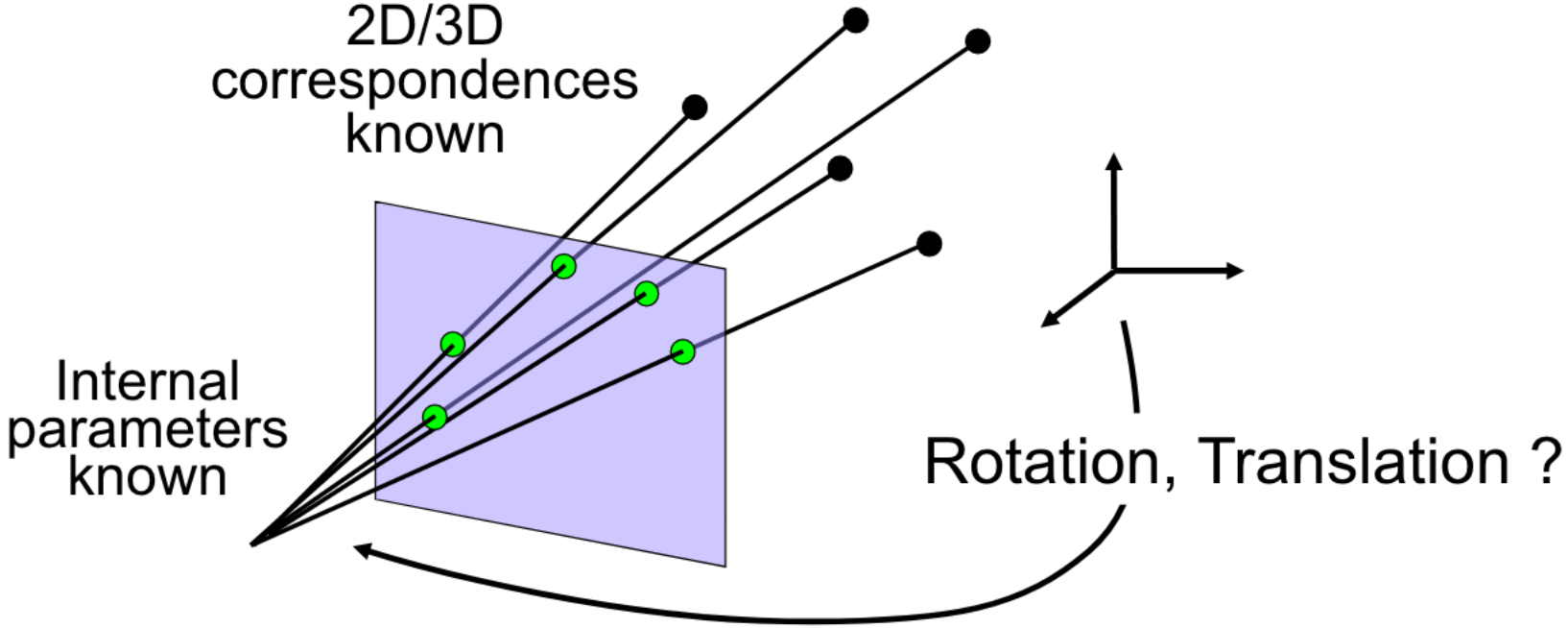
Scanning LIDAR

- Mirror scans the environment (in 2D)
- Electronics measure time between laser pulse and photodiode response



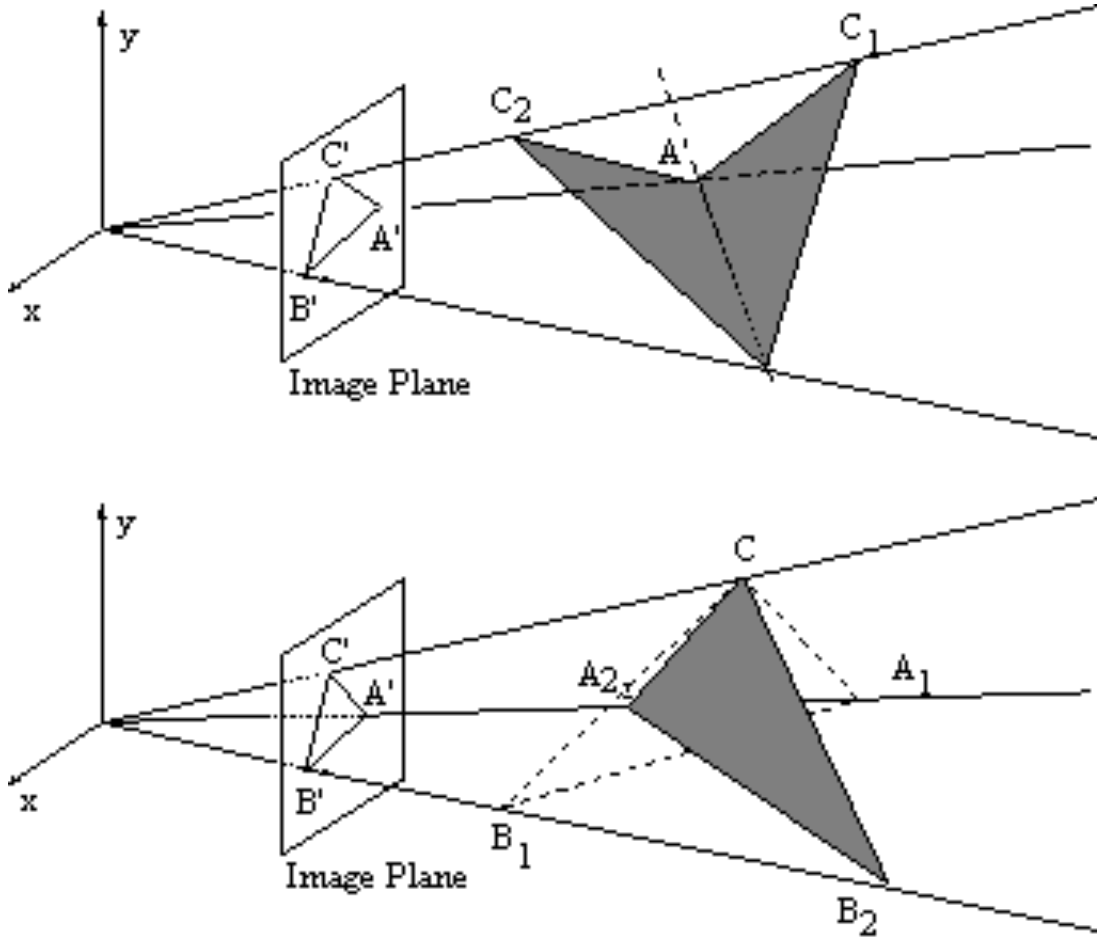
Finding the Extrinsics

- Perspective n-Point problem



P3P – Multiple solutions

- 3 Points has finite number of solutions
- Still ambiguous – 4 feasible solutions
- P4P disambiguates, but if points are co-planar in 3D, is unambiguous.
- P6P has 1 solution for 3D points



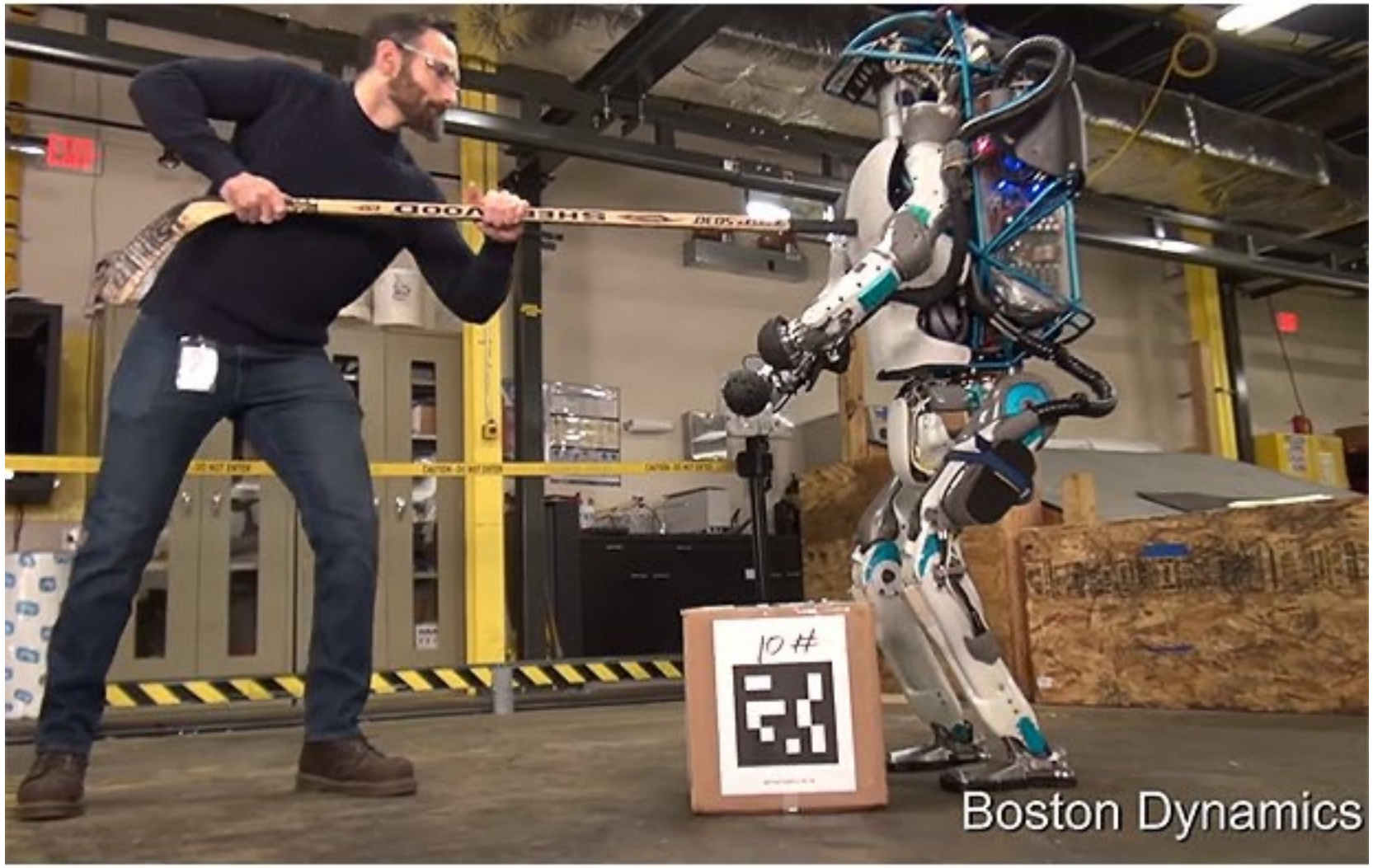
OpenCV - SolvePnP

```
(success, rot_vec, trans_vec) = cv2.solvePnP(model_points,
                                             image_points,
                                             camera_matrix,
                                             dist_coeffs,
                                             flags=cv2.CV_ITERATIVE)
```

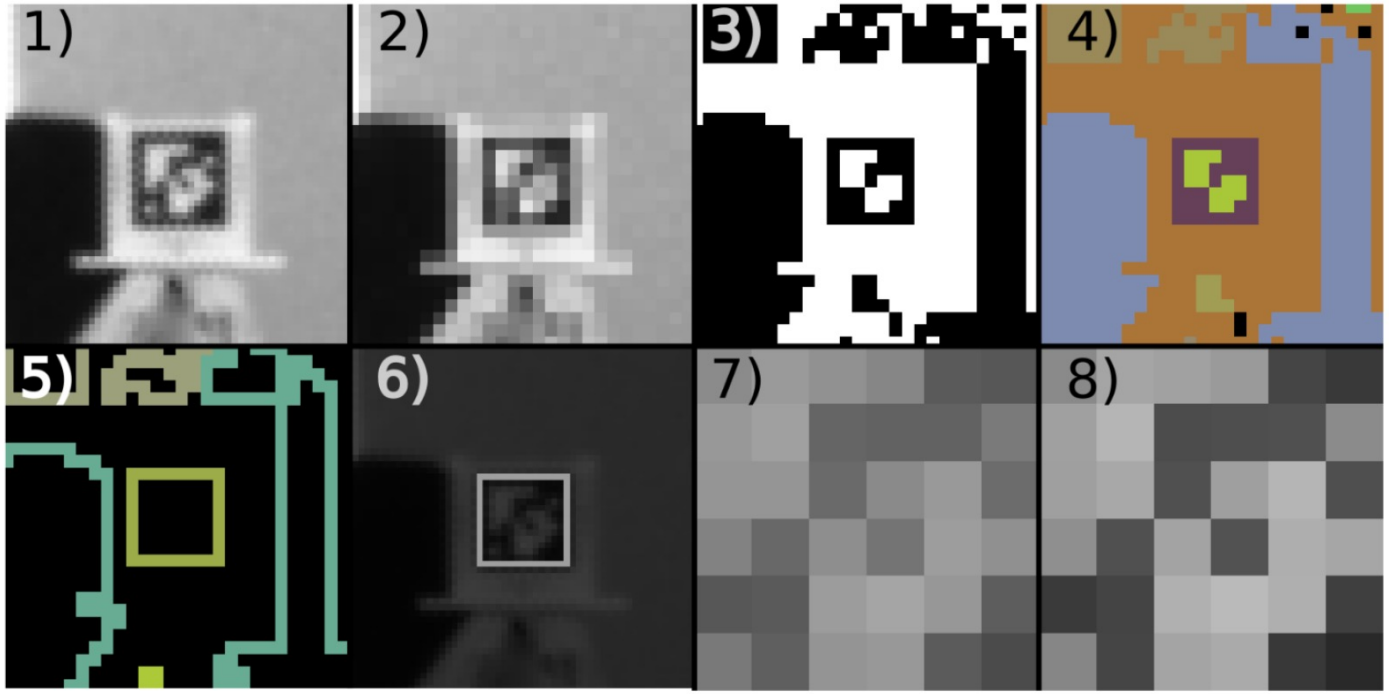
- model points - vector of N 3D points (float32)
- image_points - Array of corresponding image points (float32)
- camera_matrix – camera intrinsic matrix
- dist-coeffs – distortion coefficients
- Flags - algorithm

Apriltags

- Fiducial Markers
- April lab at UM



Apriltags – How do they work



1. Capture image
2. Decimate (down sample)
3. Threshold
4. Segment (union find)
5. Contour segmented image
6. Find quadrilaterals
7. Perspective correction of quad
8. Decode

Apriltags – How do I use it?

- Config options in /opt/ros/melodic/share/apriltag_ros/config
- Add tag IDs you want to detect to config/tags.yaml
- Set-up detector settings in config/settings.yaml
- Options: tag family, decimate, blur, refine-edges, debug
- Run apriltag_ros node, give source image topic and intrinsic calibration topic

```
$ roslaunch apriltag_ros continuous_detection.launch
camera_name:=/camera/color/ image_topic:=image_raw
```

Apriltags – what do you get?

```
apriltag_ros/AprilTagDetection[] detections
  int32[] id
  float64[] size
  geometry_msgs/PoseWithCovarianceStamped pose
    std_msgs/Header header
      uint32 seq
      time stamp
      string frame_id
    geometry_msgs/PoseWithCovariance pose
          geometry_msgs/Pose pose
                geometry_msgs/Point position
                geometry_msgs/Quaternion orientation
          float64[36] covariance
```