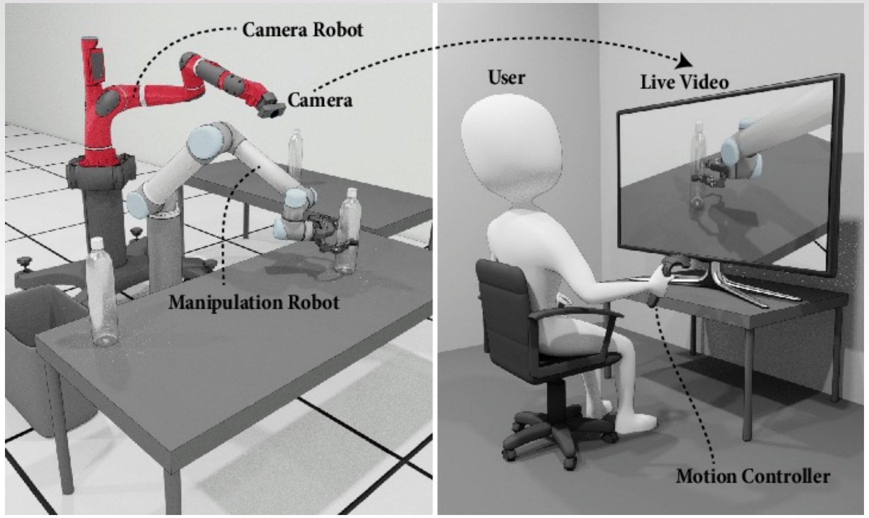
Occlusion-Free Visual Servoing for the Shared Autonomy Teleoperation of Dual-Arm Robots

Overview

- **Background** = Teleoperation becomes popular (especially during Covid periods) and we try to innovate among interdisciplinary fields regarding robot and teleoperation (e.g. tele-nursing robot)
- **Research Purposes** = Design useful interfaces applicable for tele-nursing robot which help improve user's performance during teleoperation (easy-to-use, decrease workload, etc.)
- **Challenges** =
  - 1. Visual Sensing (loss of depth via 2D image and camera views are limited)
  - 2. Haptic Sensing (Non-haptic feedbacks from robot via remote control)
  - 3. Motion Control (Need efficient motion planning algorithms to avoid sickness)

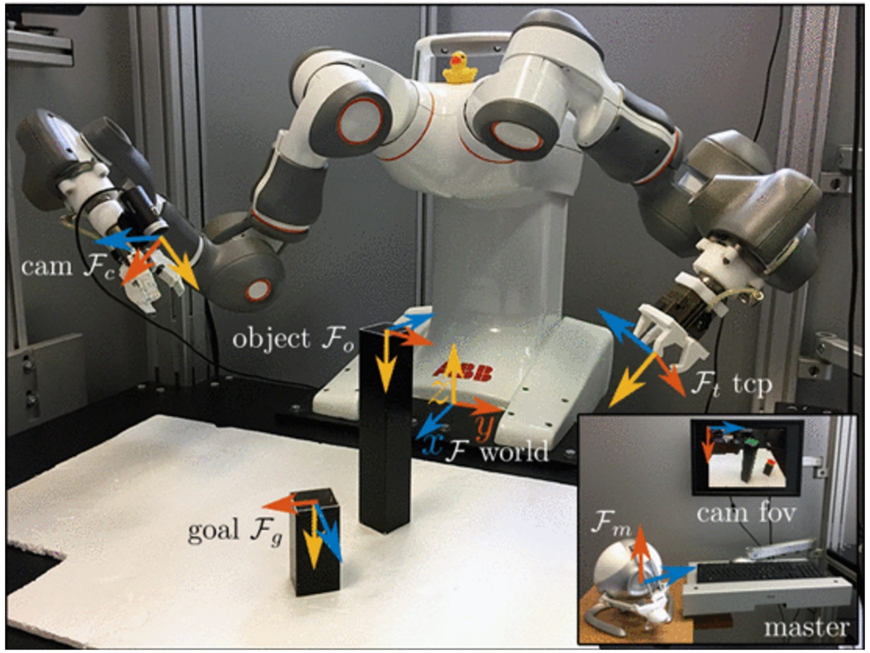
Research Question

how to use the eye-in-hand camera to provide autonomous occlusion-free viewpoint for the remote user to observe the task performed by another manipulator in a teleoperation system?

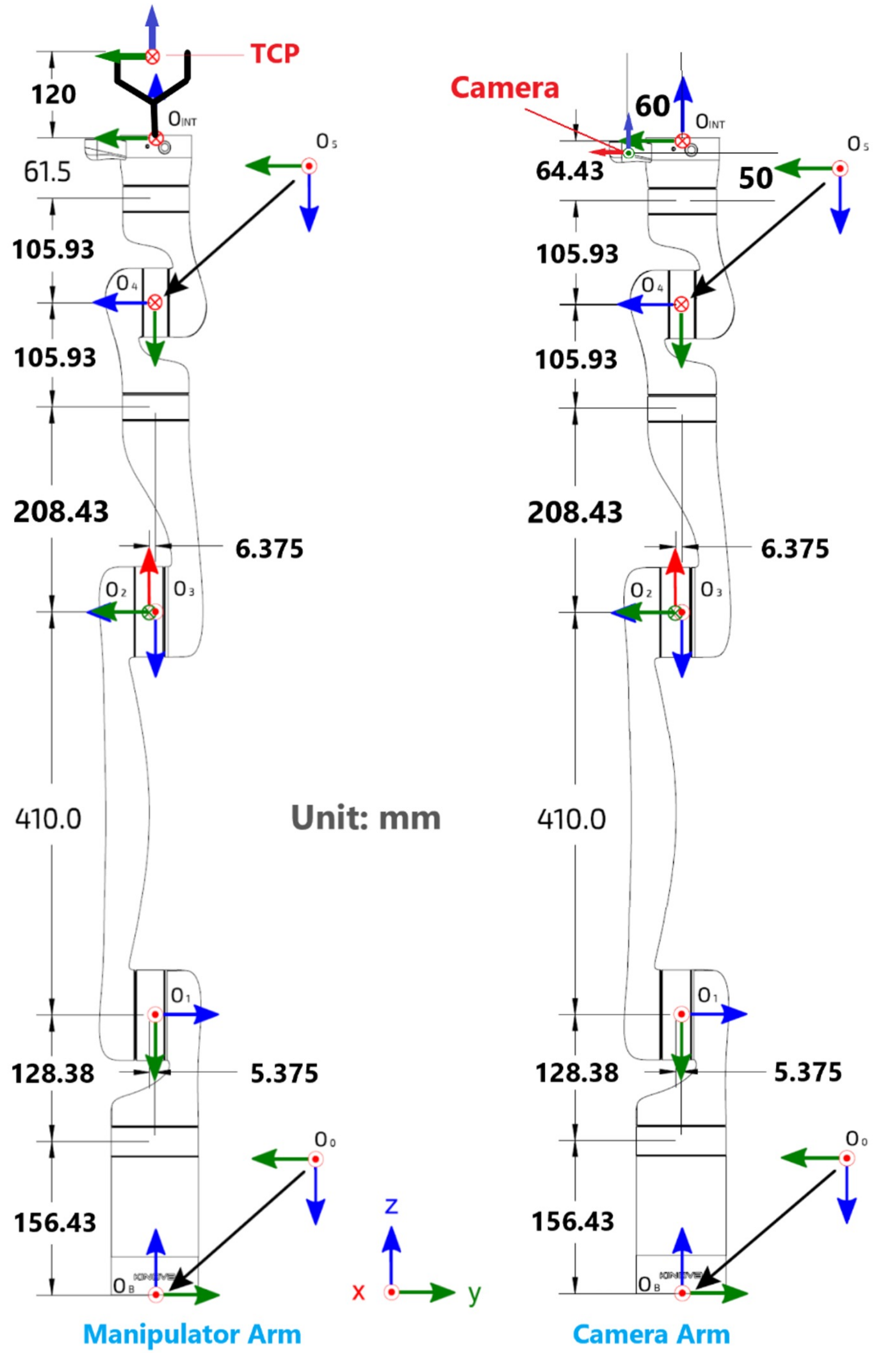


Objective

- Autonomous continuous camera positioning to avoid occlusions and obstacle collision
- Teleoperated TCP (Tool Center Point) and user-selected goal visible at all times in the camera FoV
- Natural and intuitive camera motion and reference mapping of the manipulator in the camera frame.



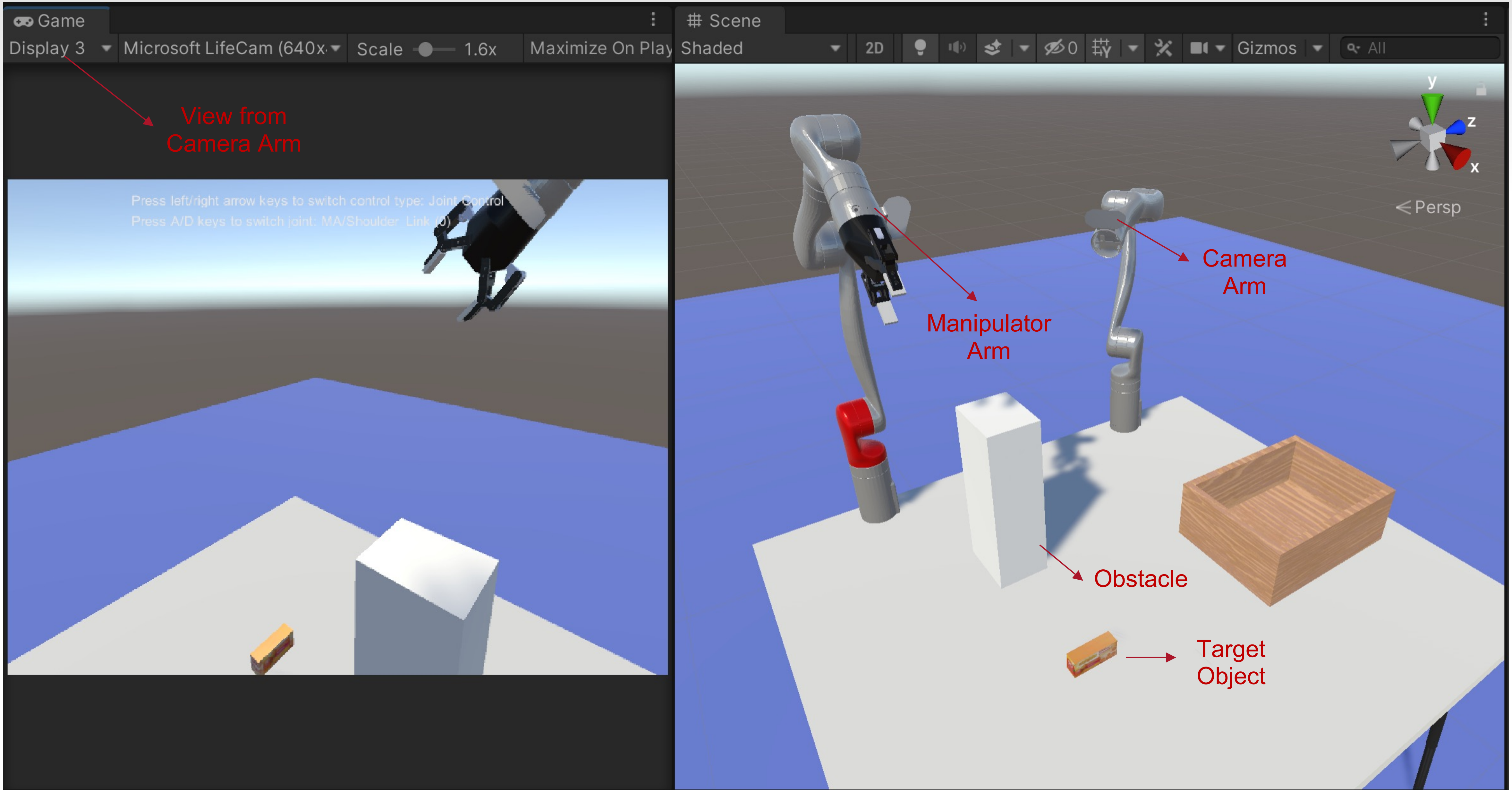
Kinova Model



Manipulator Arm				
i	θ/rad	d/m	a/m	α/rad
0 (Base)	0	0	0	π
1	q1	-0.28481	0	π/2
2	q2 - π/2	-0.005375	0.41	π
3	q3 - π/2	-0.006375	0	π/2
4	q4 + π	-0.31436	0	π/2
5	q5 + π	0	0	π/2
6 (TCP)	q6 + π	-0.28743	0	π

Camera Arm				
i	θ/rad	d/m	a/m	α/rad
0 (Base)	0	0	0	π
1	q1	-0.28481	0	π/2
2	q2 - π/2	-0.005375	0.41	π
3	q3 - π/2	-0.006375	0	π/2
4	q4 + π	-0.31436	0	π/2
5	q5 + π	0	0	π/2
6 (Camera)	q6 + π	-0.15593	0.06	π

Unity Environment Setup



IBVS (Image based visual servoing)

Camera Arm Joint Velocity $\xleftrightarrow{\text{Mappings}}$ 2D Image Pixel Velocity

tool translational velocity wrt Fc

robot camera joint velocities

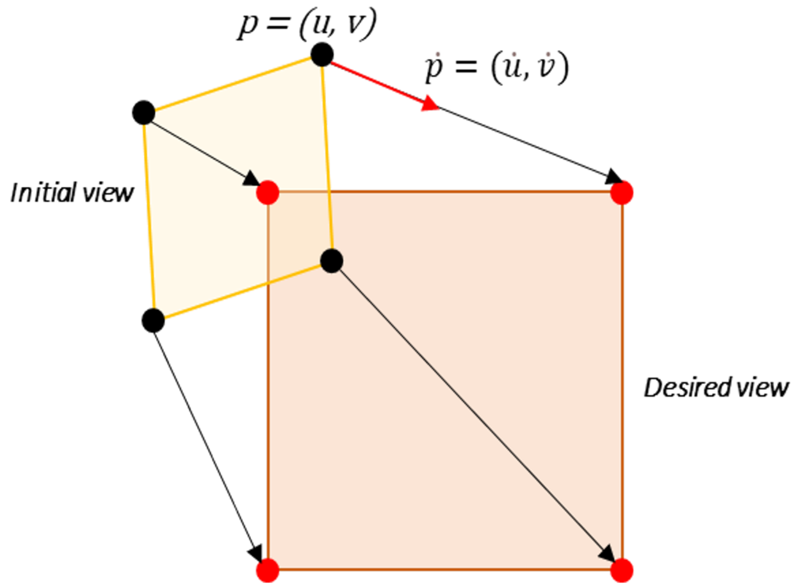
rotation matrix from world to camera frame

$$\dot{\boldsymbol{p}}_t = \boldsymbol{L}_t \begin{bmatrix} \boldsymbol{v}_c^c - \boldsymbol{v}_t^c \\ \boldsymbol{\omega}_c^c \end{bmatrix} = \boldsymbol{L}_t \boldsymbol{J}_c^c \dot{\boldsymbol{q}}_c + \boldsymbol{L}_t \begin{bmatrix} -\boldsymbol{R}^c & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{J}_t \dot{\boldsymbol{q}}_t$$

arm joint velocities

tool Jacobian in F

$$= \begin{bmatrix} \boldsymbol{P}_{c,t} & \boldsymbol{P}_{t,t} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_c \\ \dot{\boldsymbol{q}}_t \end{bmatrix} = \boldsymbol{P}_t \dot{\boldsymbol{q}}$$



joint ref vector for controller

null space projector

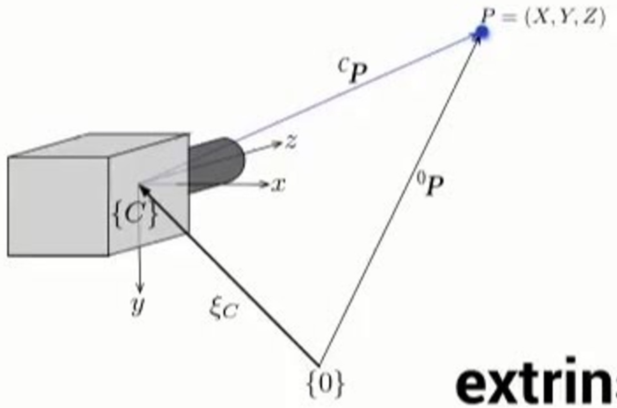
IBVS Control Law: $\dot{\boldsymbol{q}} = k_p \boldsymbol{P}_t^{\dagger} (\boldsymbol{p}_t^* - \boldsymbol{p}_t) + \boldsymbol{N}_t \dot{\boldsymbol{q}}_0$

Pt pseudoinverse

Camera Model

Complete camera model

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$



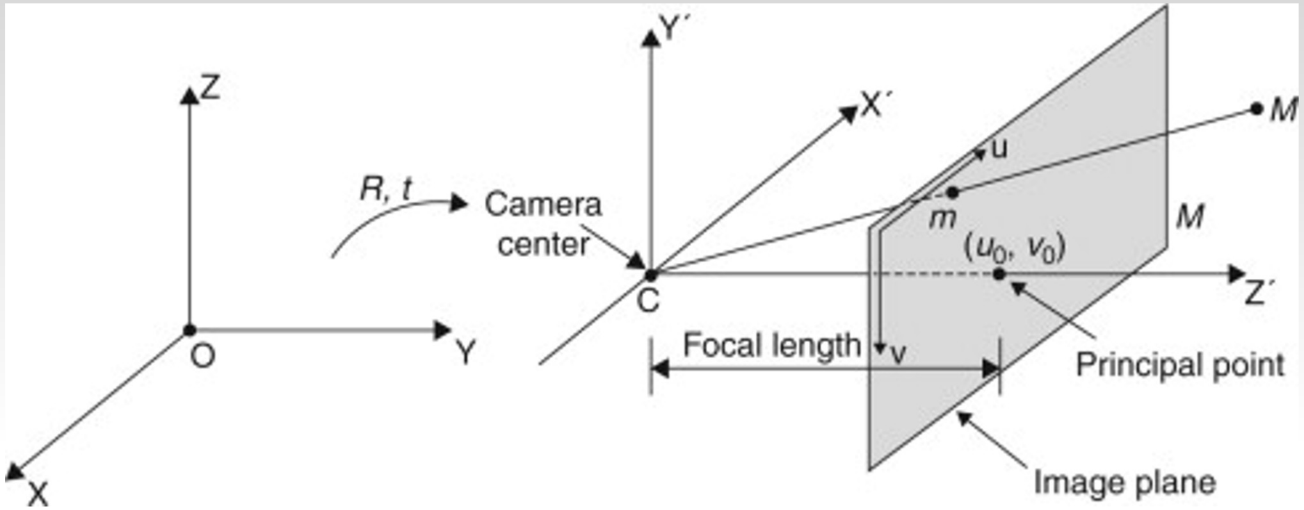
extrinsic parameters

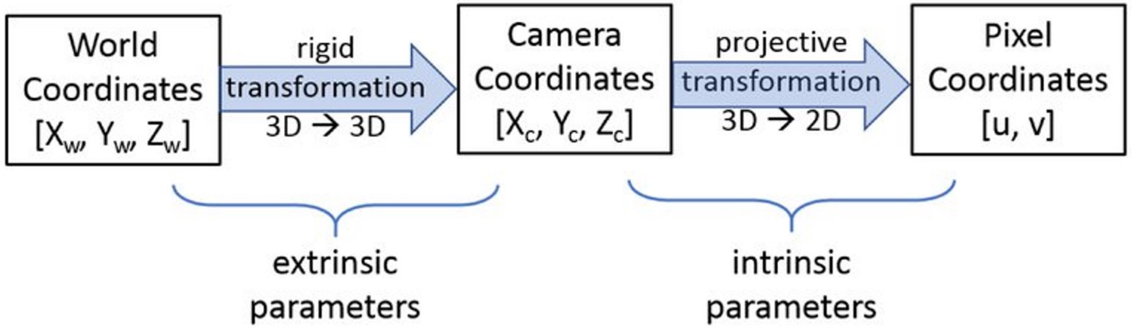
$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix} \frac{1}{\rho_u} & 0 & u_0 \\ 0 & \frac{1}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{K}} \begin{pmatrix} \mathbf{R} & t \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}^{-1}}_{\mathbf{C}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

intrinsic parameters

camera matrix

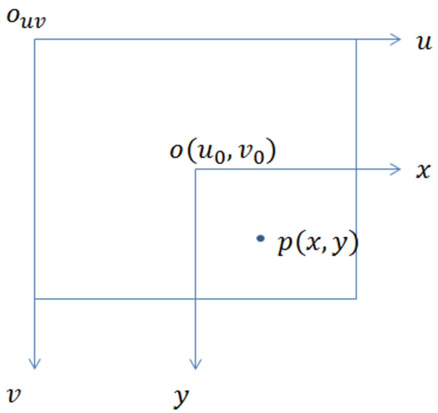
Camera Parameters





$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates | Intrinsic properties (Optical Centre, scaling) | Extrinsic properties (Camera Rotation and translation) | 3D World Coordinates



Camera Capture Resolution: 2064 x 2096 (Width x Height)

Camera Intrinsic Matrix:

1.1111	0	0.5
0	1.0941	0.5
0	0	1

Camera Extrinsic Matrix: By Implementing Camera API worldToCameraMatrix

Finite State Machine

Ss* - *Setup State:

System starting state

SA* - *Approach State:

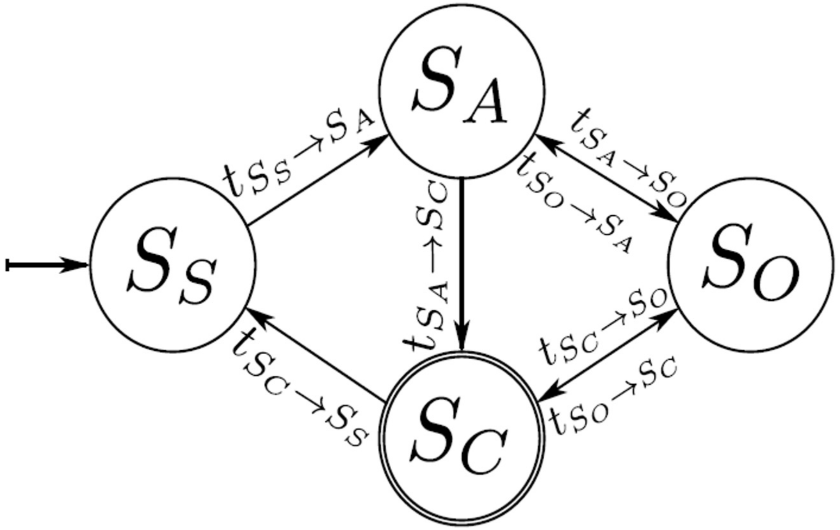
Approaching to goal, camera zoom in the scene

SO* - *Occlusion State:

Camera moves smoothly to avoid occlusion

SC* - *Conclusion State:

Free for user to reach the goal with TCP



Setup State: Ss

Cost function regulates the TCP and the COM of the goal to reference points on the image

$$e = P_{ref} - P_{next}$$

$$P_{next} = P + \dot{P}\Delta t$$

$$\dot{P} = L_p J_c^c \dot{q}$$

Cost fun.: $\|\boldsymbol{e}_t\|^2_{\boldsymbol{W}_t} + \|\boldsymbol{e}_{g,com}\|^2_{\boldsymbol{W}_{g,com}}$

Constraints: $(\boldsymbol{E}_{g,fov}, \boldsymbol{g}_{g,fov}), (\boldsymbol{E}_{lim}, \boldsymbol{g}_{lim})$

Approach State: Sa

In this state, operator uses teleoperation to move the tool manipulator arm TCP toward the goal object.

Cost function regulates the TCP and the COM of the goal to reference points on the image, as well as the fix the camera rotation in z axis to its initial state.

$$\dot{V}_c = J_c^c \dot{q}$$

$$\text{Cost fun.: } \|\boldsymbol{e}_t\|^2_{\mathbf{W}_t} + \|\boldsymbol{e}_{g,com}\|^2_{\mathbf{W}_{g,com}} + \|\boldsymbol{e}_{t,tele}\|^2_{\mathbf{W}_{t,tele}} + \|e_\phi\|^2_{w_\phi}$$

$$\text{Constraints: } (\boldsymbol{E}_{fov}, \boldsymbol{g}_{fov}), (\boldsymbol{E}_{lim}, \boldsymbol{g}_{lim}), (\boldsymbol{E}_{occ}, \boldsymbol{g}_{occ})$$

Conclusion State: Sc

In this state, the TCP of the manipulator arm is close enough to the goal, and the goal object in the image is big enough.

Calculate the area of a triangle

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

$$\text{Cost fun.: } \|\boldsymbol{e}_{g,com}\|^2_{\boldsymbol{W}_{g,com}} + \|\boldsymbol{e}_{t,tele}\|^2_{\boldsymbol{W}_{t,tele}} + \|e_{g,area}\|^2_{w_{g,area}} + \|e_{\phi}\|^2_{w_{\phi}}$$

$$\text{Constraints: } (\boldsymbol{E}_{fov}, \boldsymbol{g}_{fov}), (\boldsymbol{E}_{lim}, \boldsymbol{g}_{lim}), (\boldsymbol{E}_{occ}, \boldsymbol{g}_{occ})$$

Occlusion State: So

In this state, in order to give more flexibility to camera arm, we release the regulation of position of TCP and goal. Instead we regulate their velocity on the image.

$$\begin{bmatrix} \boldsymbol{E}_c & \boldsymbol{E}_t \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_c \\ \dot{\boldsymbol{q}}_t \end{bmatrix} \geq g$$

$$\begin{aligned} \boldsymbol{E}_c = t_b \{ & (\boldsymbol{p}_b - \boldsymbol{p}_a)^T (\boldsymbol{P}_{c,b} - \boldsymbol{P}_{c,a}) \overline{s}^2 \\ & + [(\boldsymbol{p}_b - \boldsymbol{p}_a)^T (\boldsymbol{P}_{c,a} - \boldsymbol{P}_{c,t}) \\ & + (\boldsymbol{p}_a - \boldsymbol{p}_t)^T (\boldsymbol{P}_{c,b} - \boldsymbol{P}_{c,a})] \, \overline{s} \\ & + (\boldsymbol{p}_a - \boldsymbol{p}_t)^T (\boldsymbol{P}_{c,a} - \boldsymbol{P}_{c,t}) \} \end{aligned}$$

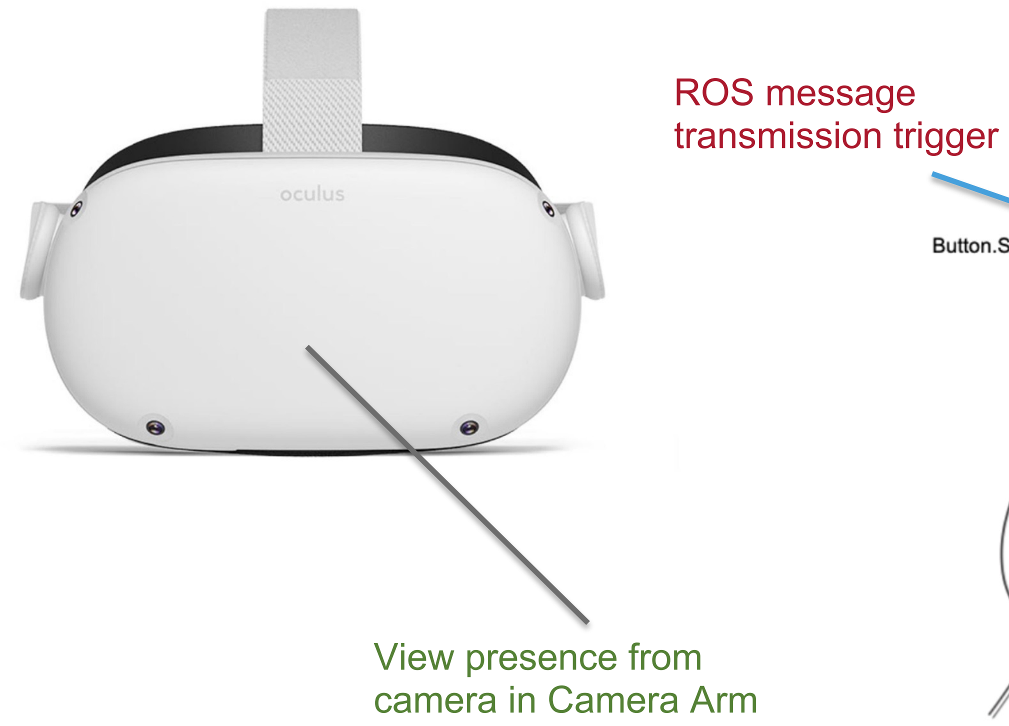
$$\boldsymbol{E}_t = -t_b \left[(\boldsymbol{p}_b - \boldsymbol{p}_a)^T \overline{s} + (\boldsymbol{p}_a - \boldsymbol{p}_t)^T \right] \boldsymbol{P}_{t,t}$$

$$g = -(\alpha \overline{s}^2 + \beta \overline{s} + \gamma)$$

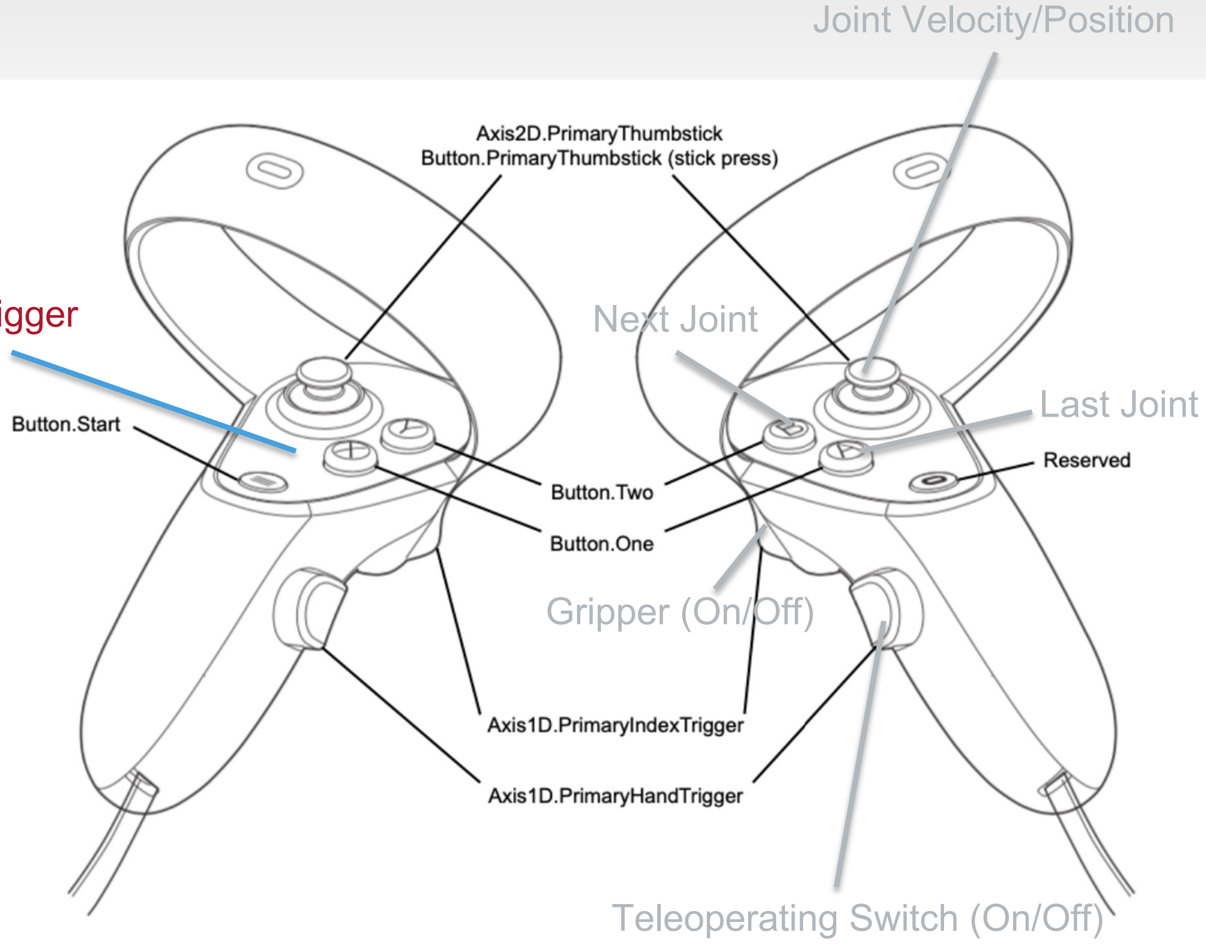
$$\text{Cost fun.: } \|\dot{\boldsymbol{p}}_t\|^2_{\boldsymbol{W}_t} + \|\dot{\boldsymbol{p}}_{g,com}\|^2_{\boldsymbol{W}_{g,com}} + \|\boldsymbol{e}_{t,tele}\|^2_{\boldsymbol{W}_{t,tele}} + \|\boldsymbol{e}_{o,com}\|^2_{\boldsymbol{W}_{o,com}} + \|a_o\|^2_{w_{o,area}} + \|v^c_{c,z}\|^2_{w_{c,z}}$$

$$\text{Constraints: } (\boldsymbol{E}_{fov}, \boldsymbol{g}_{fov}), (\boldsymbol{E}_{lim}, \boldsymbol{g}_{lim}), (\boldsymbol{E}_{occ}, \boldsymbol{g}_{occ})$$

Oculus Quest 2 VR HMD Mapping

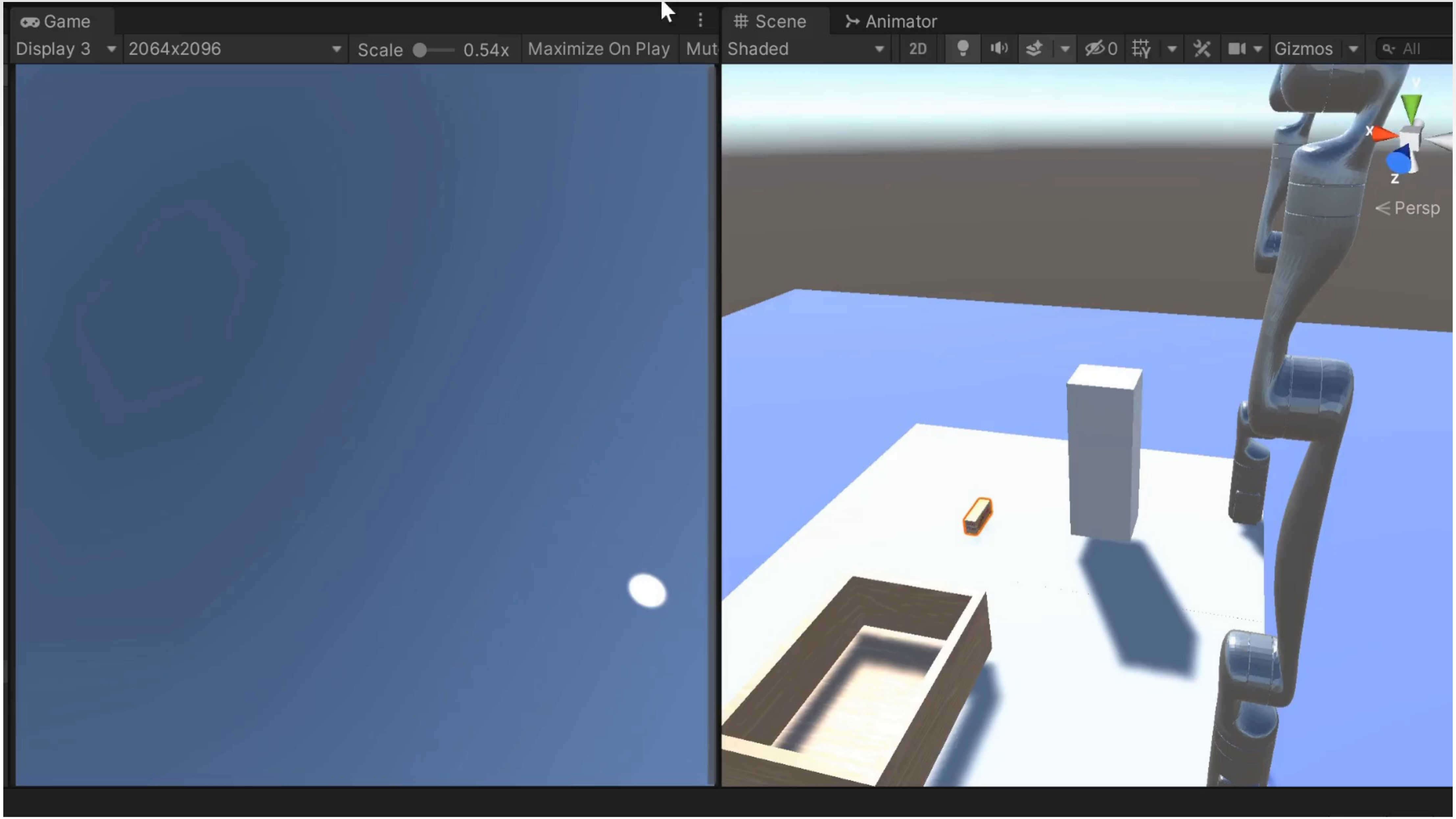


Oculus Headset

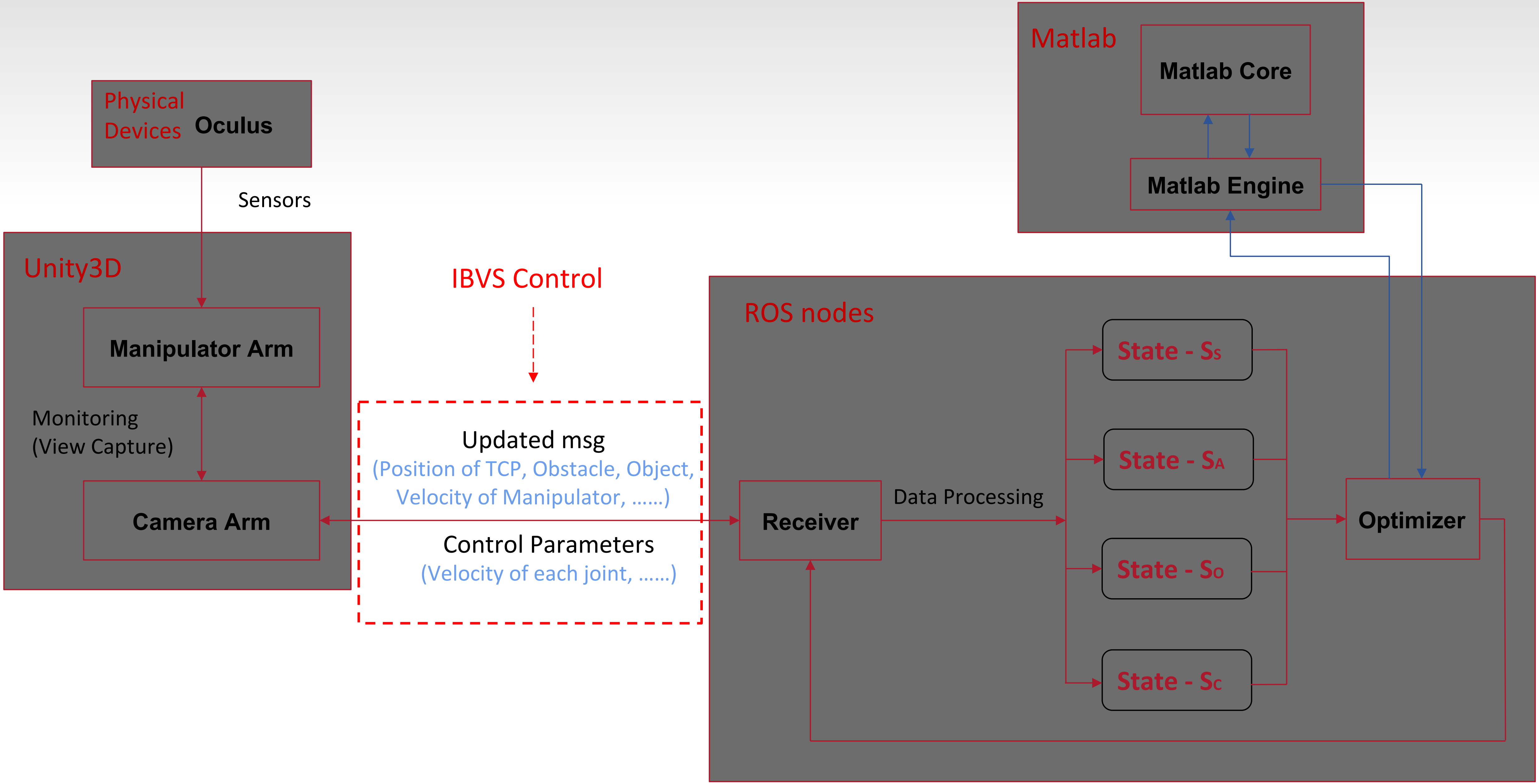


Oculus Controller

Overall Grasping Process



IBVS Unity-ROS Framework



Conclusion (Intellectual Merits)

1). This project has both Linux version and Windows version implementation

2). Oculus is one example of VR HMD manipulation for remote scene telepresence where the system also provides interfaces used for designing other similar VR devices as the hardware part

3). A framework has been created in this project where Kinova arm is utilized but it also works for other robot models with certain DH parameters given

4). Unity built-in camera can be replaced with any real camera, and Computer Vision techniques must be implemented for object recognition

5). Kinova Model could be combined with physical robot arms using hardware driver under both Linux/Windows environment

Limitations

1). Because WSL is used under Windows OS, there could be accident errors happening due to WSL limitations itself (e.g. Lack of package dependency, hardware driver problem, etc.)

2). The obstacle mapping to the 2D image must be in a convex geometric shape rather than concave one

3). The robustness of optimization hasn't been proved yet, because of which the optimized data may not be applicable under some specific situations