Question 1 IVR

1. Create a node to move the other three joints

Create a Python executable (let's call it Publicator) that publishes to the separate Joints according to the required sinusoidal signals. This could be re-used by parts 1 & 2 as the only difference is which node is "frozen".

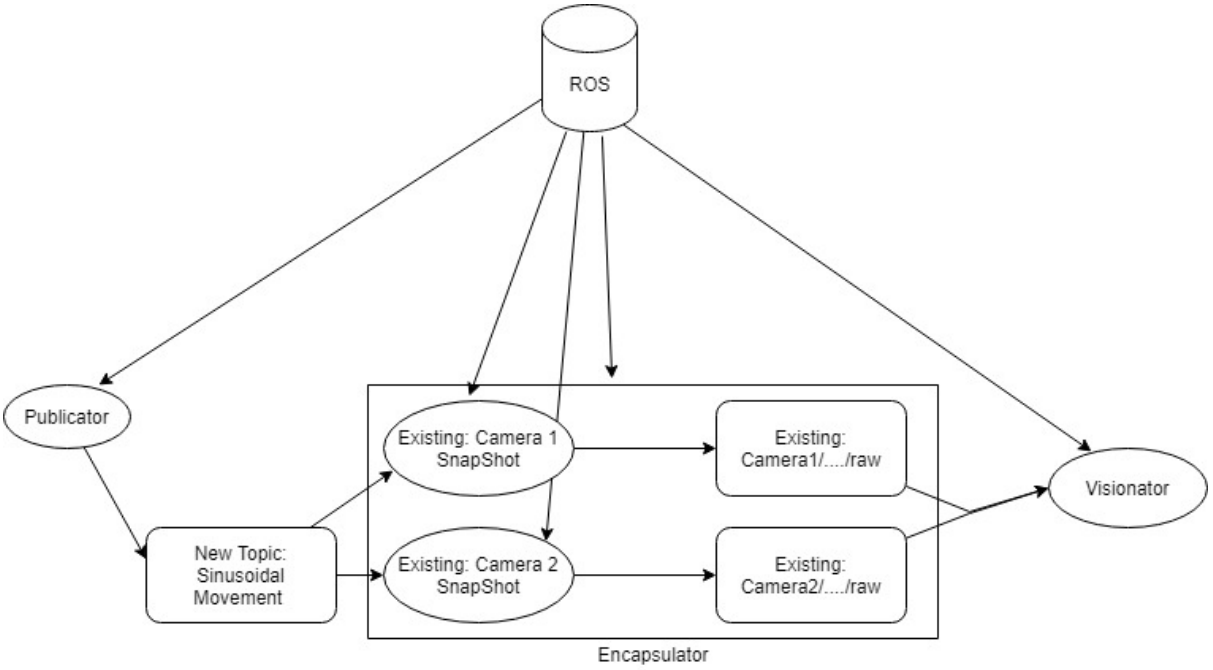
The main challenge is ensuring that the time t is consistent across all steps of the publication/consumption. I.e. the publication & movement happen in a way that they can be tracked to each other.

To illustrate this using an example, If Publicator publishes to topic "/robot/joint{x} position controller/command", it seems to take 300ms from the publication till the robot movement. This could be due to the lag between the rosrun command & its registration by ROS or could be between the receipt of the publication by the subscriber & the actual movement of the robot.

If signals are sent at a faster rate (e.g. every 50ms), there is a risk that the publication & the movement happen in a way that it is difficult to assign the correct timestamp to each movement.

2. Create a node that estimates the value of the 3 joints using computer vision

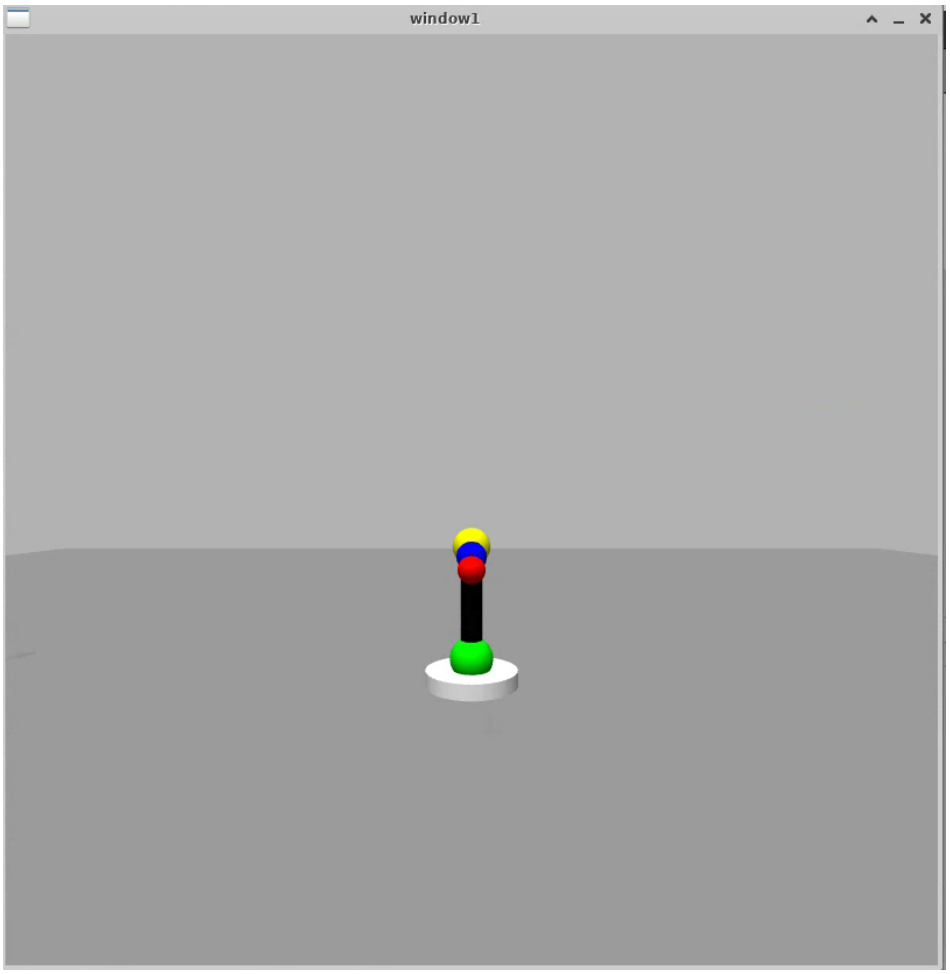
Combine the input of the two cameras to estimate the angles of the joints. Ideally, the process of information goes like this [provisional]:



The output type of the Visionator needs to be determined.

Challenges:

A. As with point 1, we will be to ensure the timestamps across the full process are aligned. It could be that this is automatically handled by ROS and it's a non-issue but we have to ensure this is the case. The timestamp t published by the Publicator must be the same t for which the camera1/raw & camera2/raw images are received by the Visionator
B. The angles may twist the image such as that the nodes are not visible by one camera alone. We will have to find a way to combine the two images. One approach would be to have a first mask where we identify which image contains more pixels of each node and use that image. Another approach would be to effectively combine the inputs for each node by transposing the 2D images into the whole x-y-z plane. An example of a dubious image from Camera1 is the following:



C. When creating the estimated vs calculated angles, again, we need to make sure that the t used is consistent. As such, it's probably a good idea to embody the timestamp to the output of the Visionator node, rather than calculating offline to avoid any risk of misalignment between calculation & estimation

The second part of Question 1 IVR seems very similar to the first part. We need to make sure whatever approach we use is compatible & re-usable.

Task Breakdown:

1. Topic creation/evaluation. Need to ensure that the existing topics are sufficient for the information transfer between the Publicator/Camera/Visionator
2. Node creation: The Publicator & Visionator nodes need to be created. For consistency sake (ahemm), I suggest we keep those names
3. Evaluate how to combine the two camera inputs to one (Encapsulator) to avoid any synchronisation issue
4. Creation of the Visionator software where vision techniques are applied to correctly identify the position of the nodes on the robot arm. Under here, many helper functions may be required such as coordinate transformation, OpenCV applications, etc.
5. Output of the Visionator should be defined in accordance to the required plots