



Project progress

Presentation to the tutors

Overview of the communication between the repositories

Issues:

- ***Different way of compilation*** (virtual environment for Tensorflow, catkin_make for ip-camera and our nodes, build for ORB_SLAM2,...)
- ***Instability on Linux*** (frequent freeze of Linux when the program is running, impossibility to re build the ORB_SLAM2, ip_camera affected by the update of Linux libraries, lot of problems with the graphic card's drivers with Tensorflow, need a lot of computer's resources : CPU, GPU ,...)

Alternative:

Instead of working on the repository, creation of a look-alike system that could be linked to the repository once they will be more stable.




Link between CloudPoint and labels

Issues:

- ***Difficult and less adaptable to send a custom class as a message*** (To publish the ORB_SLAM2 Map or the Frame for instance)
- ***Transformation between referential of the Map and referential of the image*** (The object are detected in the 2D plane of the image whereas the points of the map are in 3D)

Alternative:

Use of the sensor_msgs/CloudMap.msg for the Map. Customization of the message coming from the ORB_SLAM2 repository (Header, Array of Points and Frame). The points detected in the image are transformed in the 3D Map thanks to the frame's position and orientation.



Graphical User Interface (GUI)

Issues:

- ***OpenGL not specific for text display*** (used to display the ORB_SLAM2 cloud of points)
- ***Potential delay in the display*** (because of the computation time)

Alternative:

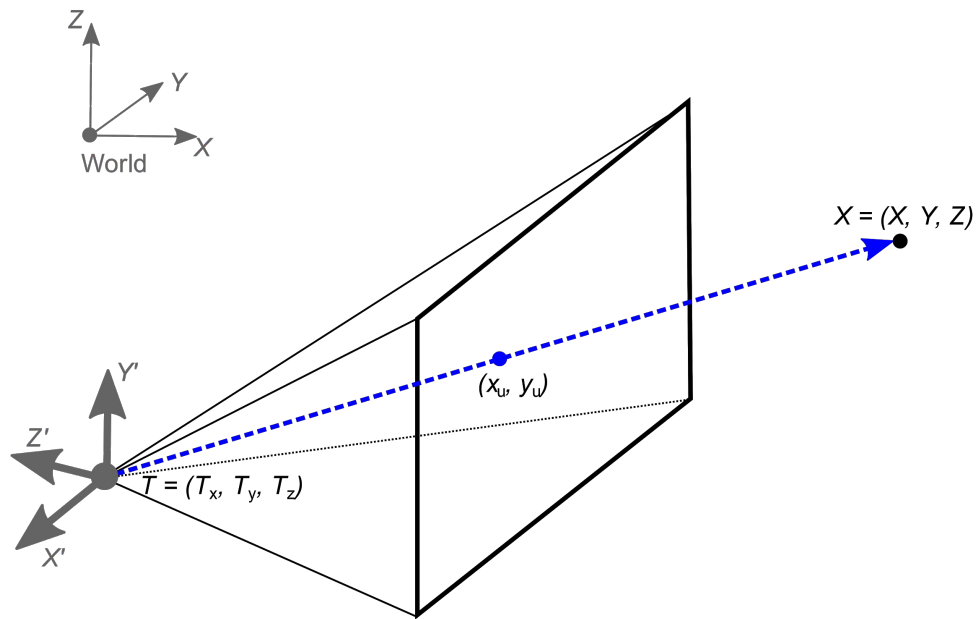
For now on no graphical interface has been implemented. What could be done is to display labels and points of interest directly on the image harvested, as a fusion between the two displays shown in the previous video



Pseudo code of the node “Labelling.py”

- **callback_Map:** get the PointCloud and the Pose of the Frame from corresponding msg
- **callback_Detection:** get the labels and bounding box for each object from corresponding msg
- **correspondingLabels:** transform the coordinate of the object from the image's referential to the global map's referential. Check for each map point if they are in a bounding box and assign the corresponding label
- **labelling:** call correspondingLabels, construct the LabelledMap msg and publish it

Extrinsic transformation



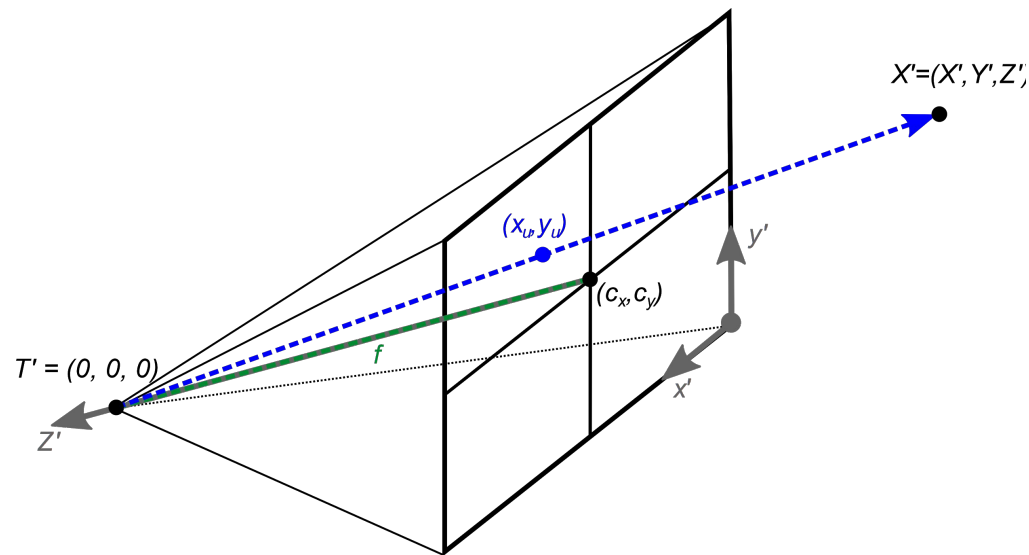
Formulas:

- Rotation Matrix
- Transposition usually needed
- $R = E + 2q_0q\sim + 2q\sim q\sim$
- $X' = R^T(X - T)$

image source:

<https://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined>

Intrinsic Transformation



Formulas:

- Focal length is virtual

$$\begin{pmatrix} x_u \\ y_u \end{pmatrix} = - \begin{pmatrix} \frac{f X'}{Z'} \\ \frac{f Y'}{Z'} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

image source:

<https://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined>