Meta data

1. Right_cam_intrinsics
2. Left_cam_intrinsics
3. Center_cam_intrinsics
4. Left_center_transform
5. Right_center_transform
6. Left_right_transform
7. Depth_intrinsics

1. Meta data is pre-computed
2. May be not all the meta data is required.
3. APIs for each meta data might help
   Eg:
   pipeline.get_right_cam_intrinsics() fetches right camera intrinsics and so on.
4. These values may be updated upon re-calibration (optional) or be pluggable.

High Level Flow

step 0 : pipeline = depthai.create_pipeline(config)

while true:

1. all_streams = pipeline.get_all_streams() #fetch all streams in config
2. center_img = all_streams['center_cam'] #get centre image if in config
3. r_img = all_streams['right_cam'] #get right image
4 l_img = all_streams['left_cam'] #get left image
5. depth_frame = all_streams['depth`]
6. Stereo_depth_frame = all_streams[‘stereo_depth’]
7. depth_aligned_to_right = pipeline.align(depth_frame, 'right_cam')
8. depth_aligned_to_center = pipeline.align(depth_frame, 'center_cam')
9. depth_aligned_to_left = pipeline.align(depth_frame, 'left_cam')
10. pointcloud = pipeline.generate_poincloud(depth_frame)
11. Inference = pipeline.infer([r_img, l_img, center_img])

Step last: depthai.delete_pipeline(pipeline)

In this setup, user experience is simple, flexible and seamless; no need to worry about - syncing, transforms, intrinsics/extrinsics, also calibration is optional.

Step 2 to 6 can vary based on user’s application. E.g may be not all users require all the streams. Similarly step 6 to 8 may vary based on user’s requirements.

pipeline.align() is used align streams. It uses meta data information to perform transformations.

Similarly pipeline.generate_pointcloud() uses meta data. The generate point cloud is smooth and compatible with common point cloud packages such as pcd, Open3D etc.