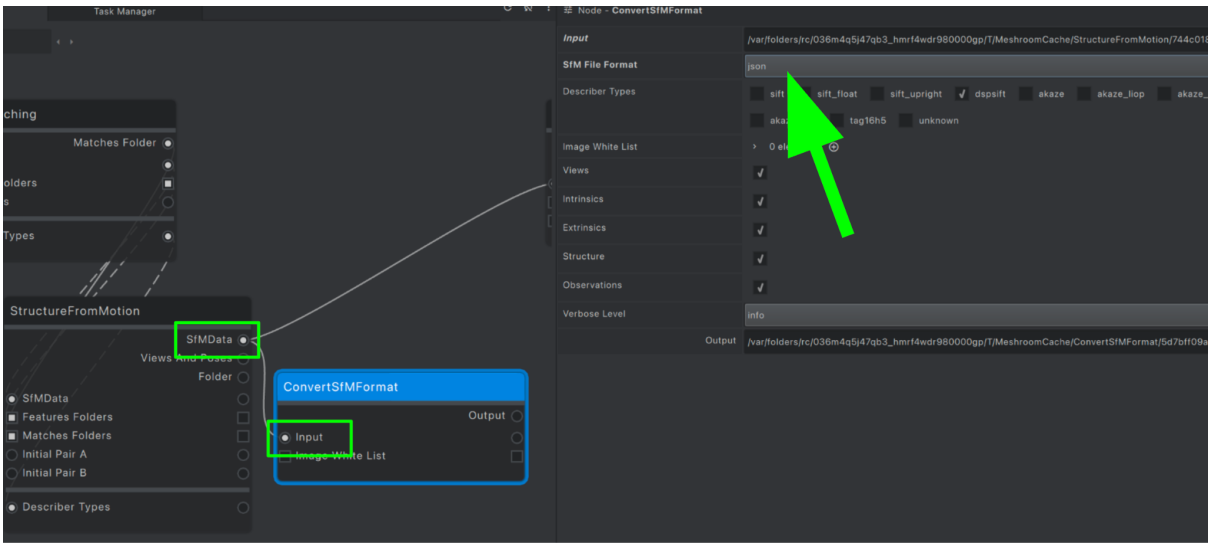
Extract camera intrinsic parameters (K and distortion coefficient) to be used in Matlab

Meshroom does not require the cameras to be calibrated, it uses the metadata of the image to initialize a first guess for the focal length during the initialization step (CameraInit node). Then it estimates the full intrinsic parameters (matrix K and distortion coefficient) during the Structure From Motion part (StructureFromMotion node).

If you want to recover the calibration parameters to be used in matlab you then have two possibilities:

- take the calibration parameters from the **StructionFromMotion node** if you managed to get a decent reconstruction (let's say 5-6 cameras at least).
  In that case, the information is stored inside the sfm.abc file that you can find in the StructureFromMotion node folder (*click right on the node and* "Open Folder"). This is a binary format that cannot be read by matlab so we need to convert it into a json format that it is easier to read.
  To do that we can use the **ConvertSfmFormat** node and add it to the pipeline to convert the file:



  Then just open the ConvertSfmFormat folder and you have the json format

- You can take it from the **CameraInit** node but in that case, you only get the focal length as reported in the metadata which is not really reliable. Opening the folder of the node (*click right on the node and* "Open Folder") you find a cameraInit.sfm that is a json file containing (among other things) the intrinsics.

In both cases, the structure of the file is a json in which the intrinsics are contained in a dictionary key "intrinsics" and a list of fields like so

```
"intrinsics": [
```

```
        {
            "intrinsicId": "759031842",
            "width": "4032",
            "height": "3024",
            "sensorWidth": "4.8899999999999997",
            "sensorHeight": "3.6675",
            "serialNumber": "mini3_Apple_iPhone 7",
            "type": "radial3",
            "initializationMode": "estimated",
            "initialFocalLength": "3.9900000000000002",
            "focalLength": "3.9900000000000002",
            "pixelRatio": "1",
            "pixelRatioLocked": "true",
            "principalPoint": [
                "0",
                "0"
            ],
            "distortionInitializationMode": "none",
            "distortionParams": [
                "0",
                "0",
                "0"
            ],
            "undistortionOffset": [
                "0",
                "0"
            ],
            "undistortionParams": "",
            "locked": "false"
        }
    ]
```


Note that:

- "focalLength" is expressed in mm, you need it in pixel format `pxFocalLength` converting it like so:

  ```
  pxFocalLength = (focalLength / sensorWidth) *
  std::max(image().Width(), image().Height());
  ```
- principalPoint is expressed as an offset wrt the center of the image, so to get the usual value you need to:
  ```
  px = image.Width() + principalPoint[0]
  py = image.Heigth() + principalPoint[1]
  ```

The K = [`pxFocalLength 0 px;  0 pxFocalLength py; 0 0 1`] and the distortions coefficients
k1 = `distortionParams[0]`
k2 = `distortionParams[1]`

```
k3 = distortionParams[2]
```