Point cloud generation Visionary devices

SICK AG | Mobile Perception
Version 1.0.0
29th of Nov. 2023

GigE Vision / GenICam 3D model

How to present camera‘s 3D output data to a generic application

- GenICam SFNC standard defines feature model to describe generated 3D data to a generic application
- https://www.emva.org/wp-content/uploads/GenICam_SFNC_v2_7.pdf (chapter 21 „3D Scan Control“)
- The data model used by Visionary devices needs extension of that model – proposal submitted to the technical committee, review (& ratification) in progress
- This presentation provides overview of the newly proposed output mode „ProjectedC“ and instructions how to build the point cloud (X/Y/Z point coordinates) from acquired data

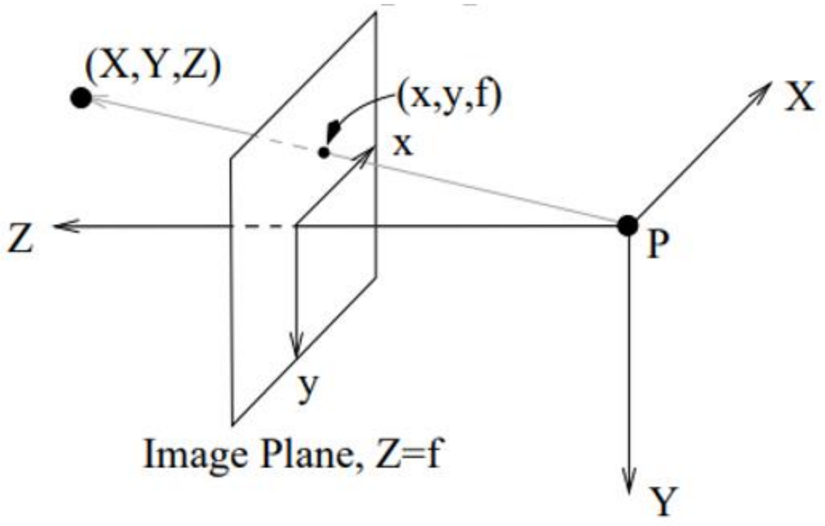
Projective camera geometry

Proposed „ProjectedC“ standard mode overview

Image formation can be approximated with a simple pinhole camera

Knowing camera intrinsic parameters, the range (Z) value itself is enough to reconstruct the other coordinates

(source: https://www.cs.toronto.edu/~jepson/csc420/notes/imageProjection)



The 3D point coordinates (X, Y, Z) for a pixel (x, y) is given by the projective transformation

$$\begin{pmatrix} x \\ y \\ f \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Considering the principal point (optical center) coordinates o_x/o_y and aspect ratio a of not-ideally-square pixels, we can calculate with camera intrinsic matrix in form

$$M_{in} = \begin{pmatrix} f & 0 & o_x \\ 0 & fa & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

Main parameters describing the 3D scene

Which features to read for each acquired frame

SICK
Sensor Intelligence.

The intrinsic parameters can be read *per-frame* through following camera „chunk" features (actual use on next slide):

```
// write (select coordinate of interest)
ChunkScan3dCoordinateSelector = „CoordinateC"
// read
scaleC = ChunkScan3dCoordinateScale
offsetC = ChunkScan3dCoordinateOffset
princPtU = ChunkScan3dPrincipalPointU
princPtV = ChunkScan3dPrincipalPointV
focLen = ChunkScan3dFocalLength
aspectR = ChunkScan3dAspectRatio
```

Notes:

- *Read through „chunk" (per-frame) features, chunk mode must be ON (ChunkModeActive)*
- *The parameters remain constant, however, non-chunk versions of the features might be introduced in the future*

Generating the point cloud

Compute all point coordinates from acquired range map

SICK Sensor Intelligence.

Given the parameters read in the previous step, the 3D coordinates corresponding to individual pixels can be calculated:

```
for (row = 0; row < imageHeight; row++)
for (col = 0; col < imageWidth; col++)
{
  xp = (col - princPtU) / focLen
  yp = (row - princPtV) / (focLen * aspectR)


  scaledC = image[row][col]* scaleC + offsetC

  xc = xp * scaledC
  yc = yp * scaledC
  zc = scaledC
}
```

Note: The calculated coordinates are given im millimeters (as also reported through ChunkScan3dDistanceUnit)

Invalid data

Identify invalid pixels

- Pixels with invalid data („no measurement“) encoded using **zero** range map value
- Ignore such pixels when calculating the point cloud (previous step)
- *(the invalid data value also reported through standard features ChunkScan3dInvalidDataFlag/ChunkScan3dInvalidDataValue)*

Reference coordinate system

Use device-specific calibration results to transform the coordinate system

- The camera outputs measurement in its device-specific „anchor“ coordinate system…
- …which can (for example due to inevitable mounting inaccuracies) differ from the ideal „reference“ coordinate system
- Reference system: right hand Cartesian, Z pointing away from the device
- Standard features to query parameters of the anchor-reference transformations (rotations and translation):
  - ChunkScan3dCoordinateReferenceSelector
  - ChunkScan3dCoordinateReferenceValue
- The features currently presented by the firmware, but **not connected** to actual calibration results
  - **Do not use** in the moment, to be finished in a **future firmware version**

Multiple data components

Acquiring RGB information together with the 3D range data

- Besides the range data, the device can also output RGB color information for each pixel
- Device features to control the component selection: ComponentSelector, ComponentEnable
- Refer to the provided sample programs (Python scripts) how to configure which components to acquire and how to identify them within the acquired data
- *Note: some older GigE Vision receivers not aware of GigE Vision „multi-part“ feature might not be able to acquire more than one component at once*