Update of the AUKinect module to allow for direct retrieval of point cloud data

by

Kristian Villien – s020320
(Jan 2012)

Introduction

To allow for faster development of plugins using 3D data, the point cloud library is to be implemented on the Mobotware. The Point cloud library works with XYZ or XYZRGB data points objects, and the 3D cameras used in the system therefore needs to be able to supply this data to the plugins.

The purpose of this document is to describe the changes made to the aukinect plugin, to allow it to supply point cloud data directly to other plugins. The data transfer has been decided to be handled by a plugin method as described http://timmy.elektro.dtu.dk/rse/wiki/index.php/Variables#Methods.

Generating 3D data

The Kinect camera device supplys a depth image, giving an 11 bit value for all pixels in the image, these values represent the coordinate in the cameras Z-plane, and not the distance to the camera lens. All conversion calculations have been taken from http://stackoverflow.com/questions/8663301/microsoft-kinect-sdk-depth-calibration.

The value is not a direct depth, but a function of the depth. A conversion from the pixel value ‘P’ to an actual Z value can be done with the equation:

$$Z = 0.1236 \cdot tan\left(\frac{P}{2842.5} + 1.1863\right)[m]$$

Finding the X and Y position can be done by simple geometry. Considering Figure 1, where x’ is the x pixel distance to the optical center, the X coordinate can be found by the equation:

$$\frac{X}{Z} = \frac{x'}{f_x} \Rightarrow X = x' \cdot \frac{Z}{f_x} \Rightarrow Y = y' \cdot \frac{Z}{f_y}$$

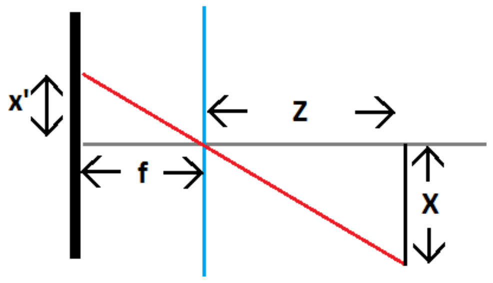

Figure 1: Finding X position

The values for focal length and optical center that are used are:

$$\cancel{f_x = 594.21434211923247}$$
$$\cancel{f_y = 591.04053696870778}$$
$$\cancel{c_x = 339.30780975300314}$$
$$\cancel{c_y = 242.73913761751615}$$

$$f_x = 595$$
$$f_y = 595$$
$$c_x = \frac{640}{2} = 320$$
$$c_y = \frac{480}{2} = 240$$

(values updated at the request of Christian Anders)

Added interface and functions

Look up tables

To speed up calculation of correct depth and XY coordinates, three look up tables have been added. The look up tables are initialized in the "init" function.

- t_depth – Holds the conversion values for every possible Kinect output to a corrected depth in mm. It should be noted that anything above 10 meters is cut off because the non linear conversion algorithm does not allow for it.
- t_posX – Holds a factor for a given X pixel that should be multiplied to the distance value to give the X position.
- t_posY – Holds a factor for a given Y pixel that should be multiplied to the distance value to give the Y position.

varUseCorrectedDepth

The variable "varUseCorrectedDepth" was added, and is by default set to 0. Setting this variable to 1 will calculate a corrected depth image and save it in the image pool with the image number saved in the 5th variable of "varImagesC3D". The data in the image is 16 bit integers with the value meaning distance in mm. Non-valid distances are set to 0 and should be filtered away later on.

It should be noted that when this variable is set, the 8bit distance image is generated using the corrected depth values. As the value of the corrected image is generally higher than the uncorrected, the value in "varUseDepth8bit" should most likely be increased.

varMakePCLFile

This variable is set to generate a point cloud file (which has the file extension .pcd) that can be read using the standard IO functions in the point cloud library. Setting this variable only makes one file using the ***next*** updated corrected depth image, so it does not work if the "varUseCorrectedDepth" is not set. Since only one file is made, the variable is automatically cleared once the file has been created. The file name is "KinPCL*.pcd" where the * is the frame number.

The .pcd file has a size of more than 8MB if all pixel values are valid. Reading and writing these files are rather slow and it is not recommended to use them unless there is no other way to get or store the data.

It is possible to make .pcd files that use binary data, however with the purpose the .pcd files currently serve, being able to read and edit them with a text editor is more important than reducing the file size and increasing load speed.

varMaxUsedDepth

This function indicates at within what distance the kinect data is valid. The default value is set to 10 since the conversion algorithm cuts it off at this distance anyway. This variable is only used for generating the point cloud data and not for the corrected distance image.

The accuracy of the kinect data falls exponentially with distance, therefore data over a certain distance is set to being invalid. Having low accuracy points in the point cloud data can cause the algorithms to use increased processing power and in some cases fail, it can therefore be a good idea to filter out these points as early as possible. This limit is used when creating .pcd files and when getting the point cloud data.

GetPointCloud (Method)

This method receives a pointer to a point cloud and fills it up with data from the latest kinect depth image. The method receives two input variables, the first indicating what kind of point cloud is used and the second indicating if there is a maximum pointcloud size (in points), with -1 meaning no limit.

Point cloud types:

- 0 = XYZ
- 1 = XYZRGB

If no limit is put on the point cloud, the maximum amount of points, if all points are value, is 480*640 = 307200 points.

The function can be called with the example code:

```
pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new pcl::PointCloud<pcl::PointXYZ>);
double pars[2], v;
bool isOK;
int n = 1;
pars[0] = 0;
pars[1] = -1;
isOK = callGlobal("kinect.GetPointCloud", "dd", NULL, pars, &v, (UDataBase**) &cloud, &n);
```

Changed interface and functions

varImagesC3D

The size of the variable “varImagesC3D” has been increased to include an imagepool number for the corrected depth image. The default value is 22, resulting in kinect images being place at 18-22 by default.

8 bit distance image

When the “varUseCorrectedDepth” is set the 8 bit distance image is generated using the corrected distance. As the value of the corrected image is generally higher than the uncorrected, the value in “varUseDepth8bit” should most likely be increased.

To do

Calibration

Currently the calibration parameters such as focal length and optical center is hard coded into the function “processDepthImages” and “GetXYZCoordinates”.

After reading some of the articles on http://openkinect.org it does not appear as if a calibration is needed, however the used parameters was found at http://stackoverflow.com/questions/8663301/microsoft-kinect-sdk-depth-calibration where some form of calibration has been done.

If it should be possible to do a calibration of the Kinect, these values should not be hardcoded, remember to do the correction in both functions.

Point cloud RGB values

In this implementation the returned point cloud with RGB data has been implemented, however it has not been tested. The relationship between RGB data and distance data has been assumed to be a direct translation between the RGB image and the depth image, however some form of corrected relationship should properly be found.

PCL Files with RGB data

Currently it is only possible to generate .pcd files with XYZ data, if XYZRGB data is needed in a .pcd file, should be implemented.

Point cloud data with time stamp

The point cloud object does currently not have a time stamp attached to it, meaning that any time information should be gotten from the image pool from the image used to generate the point cloud.

The point cloud object class does not have a timestamp variable, so either a different means of attaching a time stamp and/or frame count should be made or a new point cloud class should be constructed.

http://docs.pointclouds.org/1.0.1/structpcl_1_1_point_x_y_z.html
http://pointclouds.org/documentation/tutorials/adding_custom_ptype.php#adding-custom-ptype