IFM O3X101 Camera Documentation

Amandine FORTIER

April 2020

Contents

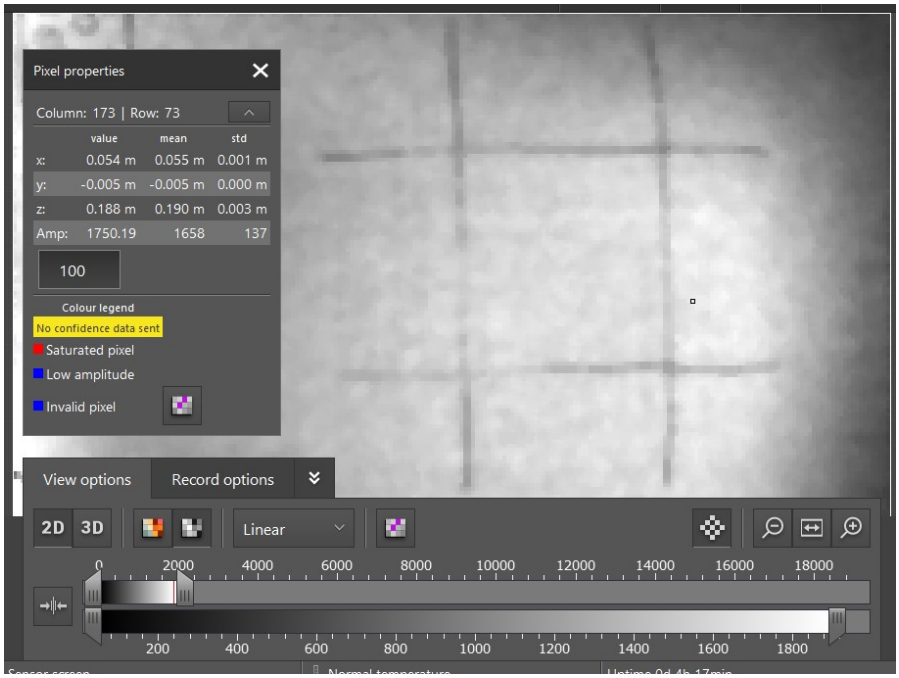
Chapter 1

Introduction

The camera is used to recognize different kinds of shapes (with the amplitude image) and detect their size (height, length and width with distance image).

The camera was connected via an RJ45 adapter to a laptop with Ubuntu 18.04. A virtual environment with Python3 was used. CLion was used for C++.

There is a software on Windows to visualize the camera's view. It looks like this.

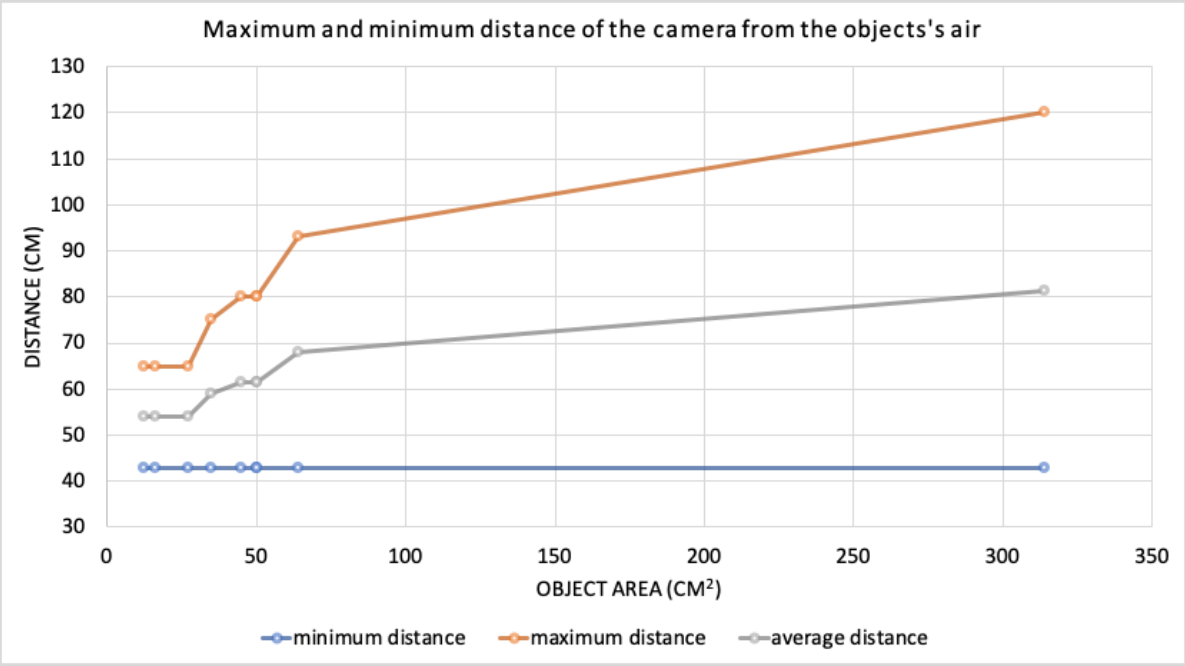


1.1 Camera distance

To have a good visibility, the minimum distance between the camera and any object is 43cm.
The maximum distance depends on the object's area. If the object's area is larger, the camera may be further away.

e.g., To correctly see an object with an air of $50cm^2$ the camera should be at a distance of 43 to 80cm.

area(cm2)	minimum distance (cm)	maximum distance (cm)	average distance (cm)
13	43	65	54
16	43	65	54
27	43	65	54
35	43	75	59
45	43	80	61,5
50	43	80	61,5
64	43	93	68
314	43	120	81,5



Chapter 2

Installation

For Ubuntu 18.04, no ROS. Every steps are on : GitHub.

2.1 ifm3d

2.1.1 Configure the ifm apt server

This step is to set up your computer to accept software from ifm's apt server.

```
sudo sh -c 'echo "deb [arch=amd64] https://nexus.ifm.com/repository/
ifm-robotics_ubuntu_bionic_amd64 bionic main" > /etc/apt/sources.list.d/
ifm-robotics.list'

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
8AB59D3A2BD7B692
```

2.1.2 Install

2.1.3 Explanations

Now that the software has been accepted from the ifm's apt server, the update has to be done to have access to the ifm's library.

2.1.4 Commands

```
sudo apt update
sudo apt install ifm3d-camera \
```

```
                    ifm3d-framegrabber \
                    ifm3d-swupdater \
                    ifm3d-image \
                    ifm3d-opencv \
                    ifm3d-picclient \
                    ifm3d-tools \
                    ifm3d-python \
                    ifm3d-pcl-viewer \
                    ifm3d-python3
```

2.2 ifm3dpy - Python

2.2.1 Explanations

First, the computer has to be up to date.
Next, all the dependencies of the ifm library have to be installed.
Then, the libraries to be installed have to be cloned. pybind11 is also a dependency. But it has to be built by hand.
After, the pybind11 dependency has to be built.
Finally, the requirements who will install ifm3dpy library has to be installed.

2.2.2 Commands

```
sudo apt update && sudo apt upgrade
sudo apt  install -y libboost-all-dev git jq libcurl4-openssl-dev \
                              libgtest-dev libgoogle-glog-dev  \
                              libxmlrpc-c++8-dev libopencv-dev \
                              libpcl-dev libproj-dev \
                              build-essential coreutils cmake python3-pip
cd lib/
git clone https://github.com/ifm/ifm3d.git
git clone https://github.com/pybind/pybind11
cd pybind11
mkdir build
cd build
cmake -DPYBIND11_TEST=OFF ..
make
sudo make install
cd ../../..
pip3 install -r requirements.txt
```

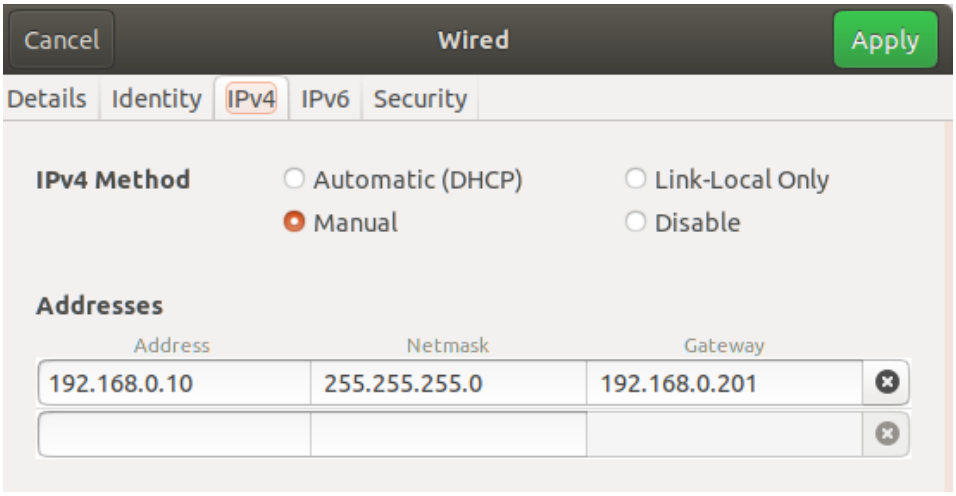
Chapter 3

Configuration

The camera can be configured as needed. Most of the time it works with the default configuration.

3.1 IP

It's an ethernet camera and its default IP is : 192.168.0.69
To communicate with it, it has to be in the same network. Because 2 devices that are not in the same network cannot communicate with each other. Gateway : 192.168.0.201



3.2 ifm3d Commands

The camera can be control and configure from the terminal.
To display help : ifm3d –help
e.g., ifm3d dumb display the sensor state.

3.2.1 Editing the configuration file

- Save the configuration file

  ```
  ifm3d dump > conf.txt
  ```

- Editing the configuration file (conf.txt)
- Load the configuration

  ```
  cat conf.txt | ifm3d config
  ```

- Check the loaded configuration

  ```
  ifm3d dump
  ```

Chapter 4

Programmation

The semantic of the ifm3d library is identical in C++ and Python.
The FrameGrabber can have several parameters depending on the image we want.

- IMG_AMP : For amplitude image.
- IMG_RDIS : Radial distance for the distance image.

4.1 C++

4.1.1 CMakeLists

```
cmake_minimum_required(VERSION 3.5)
cmake_policy(SET CMP0048 NEW)

project(cam_test CXX)

# Global compiler flags
set(CMAKE_BUILD_TYPE Release) # Release or Debug
set(CMAKE_CXX_EXTENSIONS OFF) # OFF -> -std=c++14, ON -> -std=gnu++14
set(CMAKE_CXX_STANDARD 14)
set(CMAKE_CXX_STANDARD_REQUIRED true)

if(FORCE_OPENCV3)
  find_package(OpenCV 3 REQUIRED)
elseif(FORCE_OPENCV2)
  find_package(OpenCV 2.4 REQUIRED)
```

```
else()
  find_package(OpenCV REQUIRED)
endif()

#################################################################
## finding the ifm3d lib.
#################################################################
find_package(ifm3d 0.11.0 CONFIG
  REQUIRED COMPONENTS camera framegrabber image
  )

add_executable(cam_test cam_test.cpp)
target_link_libraries(cam_test
                      ifm3d::camera
                      ifm3d::framegrabber
                      ifm3d::image
                      ${OpenCV_LIBRARIES}
                      )
```

4.1.2 cam_test.cpp

Code to save an amplitude image into "amplitude.png".

```
#include <opencv2/opencv.hpp> //for imwrite
#include <ifm3d/camera.h>
#include <ifm3d/fg.h>
#include <ifm3d/image.h>

    int main(int argc, const char **argv)
    {
        auto cam = ifm3d::Camera::MakeShared();
        auto img = std::make_shared<ifm3d::ImageBuffer>();
        auto fg = std::make_shared<ifm3d::FrameGrabber>(
                cam,ifm3d::IMG_AMP);

        if (! fg->WaitForFrame(img.get(), 1000))
        {
            std::cerr << "Timeout waiting for camera!" << std::endl;
            return -1;
        }
        imwrite("amplitude.png",img->AmplitudeImage());
```

```
        auto dist = img->DistanceImage();
        //Do something with dist
        return 0;
    }
```

4.2 Python

Code to save an amplitude image into "amplitude.png".

```
import sys
import ifm3dpy
import cv2

def main():
    cam = ifm3dpy.Camera()
    fg = ifm3dpy.FrameGrabber(cam,ifm3dpy.IMG_AMP)
    im = ifm3dpy.ImageBuffer()

    while True:
        if not fg.wait_for_frame(im, 1000):
            print("Timeout waiting for camera!")
            return -1

            cv2.imwrite("amplitude.png",im.amplitude_image())
            dist = img.distance_image()
             #Do something with dist

if __name__=='__main__':
    sys.exit(main())
```

Chapter 5

Different types of images

There is many types of images but only three of them will be explained.
An image is a 2D array. Every cell is a pixel.

5.1 Amplitude

In a greyscale picture (such as amplitude image) each pixel has a value between 0 and 255 to define its brightness.

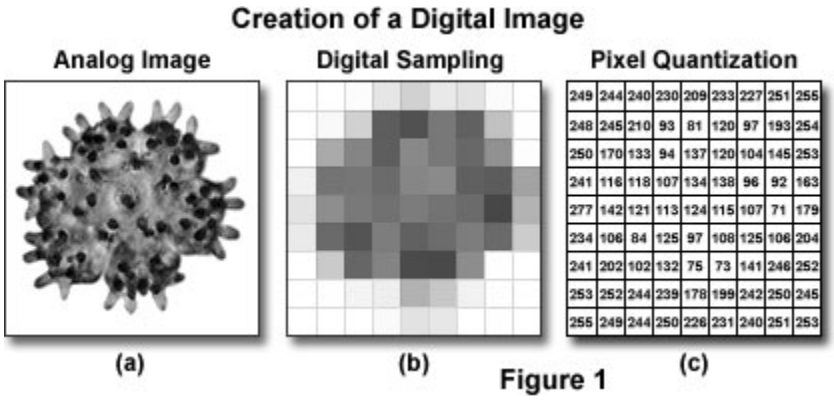


Figure 5.1: Source : http://hamamatsu.magnet.fsu.edu/articles/digitalimagebasics.html

5.1.1 Get with ifm3d library

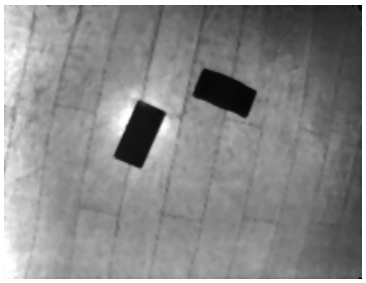
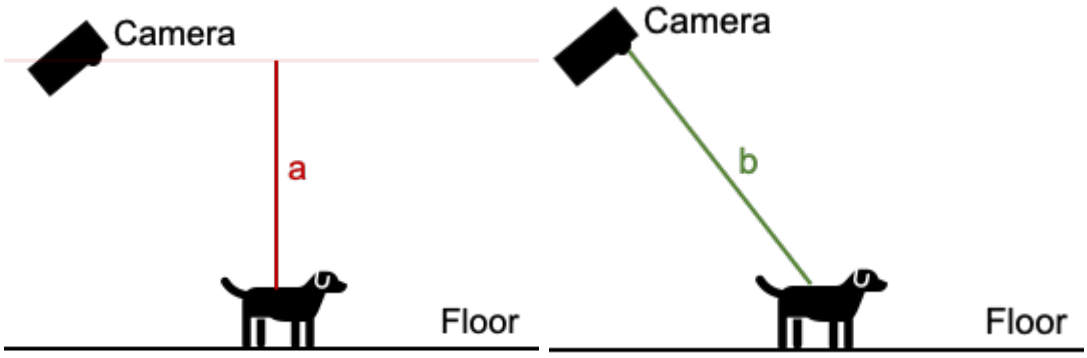


Figure 5.2: Two black boxes on the floor

5.2 Distance

In a distance image, the value of each pixel is the distance between that pixel and the camera lens. The value is in meters.
Here, each pixel contains only the distance value and not the brightness.
e.g, The distance of the pixel on the back of the dog is not "a" but "b".



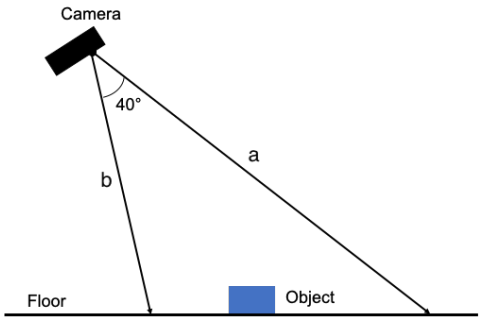
Chapter 6

Usage

There are a lot of possibilities with distance image.
The height of an object can be detected (floor distance - object distance).

6.1 Detect object's length and width

The width of the whole picture can be easily fond because we have the angle of the camera (40) and the distance from the camera and the picture's edges (a and b).



- Calculate the picture size in meters with the law of cosines $c^2 = a^2 + b^2 - 2ab\cos\gamma$
- Get the distance between the picture's edges (a and b)
- Divide the picture size by its number of pixels. Now we have the metric size of a pixel.
- Multiply the length of the object in pixels by the pixel's metric value