Name:

Partner:

Lab 1 – Distance Sensing with Computer Vision

1. **OBJECTIVE**
   1. *Lab Objective*

Students will be introduced to the field of computer vision with this lab in the form of distance sensing. Firstly, students will examine the Microsoft Kinect sensor, a valuable tool that is commonly used in computer vision and image processing applications. Next, using Microsoft Visual Studio (a C++ IDE) and OpenCV (open source computer vision libraries), students will use the Kinects to measure distances just by clicking on their computer screens.

* 1. *Key Concepts*
     1. Infrared Sensing
     2. IDE
     3. Computer Vision
     4. CMOS Sensor

1. **PROCEDURE**
   1. *Hardware and Software Setup*

Download “Computer\_Vision.zip” from ELMS and extract the folder within. Open Visual Studio and create an account with your **@umd.edu** address (do not include terpmail), if you do not have a Visual Studio account already.

In the Start Page, select the option “Open Project/Solution” and open Computer\_Vision.sln. In your Solution Explorer on the right side of the screen, expand Source Files and replace lab3\_skeleton.cpp with Lab\_1\_skeleton.cpp.

**Complete the following and answer corresponding questions in the lab:**

**Question 1:** What is OpenCV? What are some things that you can do with OpenCV?

* 1. *Depth Measurement*

You need to build Computer\_Vision.sln for every update in your code. To build a solution, select “Build Solution” from the “Build” tab. Make sure to close the .exe application

Run the application (the .exe file) under x64/Debug and measure an object at varying depths. Record the raw value output (the result variable in the code) in the table below. If the Kinect spits out a 0 for any of the depths below, ignore that row and move to the next depth.

|  |  |
| --- | --- |
| Actual depth | Depth value from Kinect |
| 40 cm |  |
| 50 cm |  |
| 60 cm |  |
| 70 cm |  |
| 80 cm |  |
| 90 cm |  |
| 100 cm |  |

**Question 2:** What is the relationship between depth value from Kinect and the actual depth in cm? Perform a fit of your data using Excel, MATLAB, or some other software.

Write a line of code so that instead of depth value, the console outputs depth in centimeter. You need to remove the folder “Computer\_Vision.tlog” under x64/Debug before building a new solution.

**Question 3:** Include your code and a screenshot of console showing cm display here.

Test out how accurate your depth calibration is by taking the same measurements where the console output is now **in centimeters**.

|  |  |  |
| --- | --- | --- |
| Actual depth | Depth value from Kinect (cm) | % Difference |
| 40 cm |  |  |
| 50 cm |  |  |
| 60 cm |  |  |
| 70 cm |  |  |
| 80 cm |  |  |
| 90 cm |  |  |
| 100 cm |  |  |

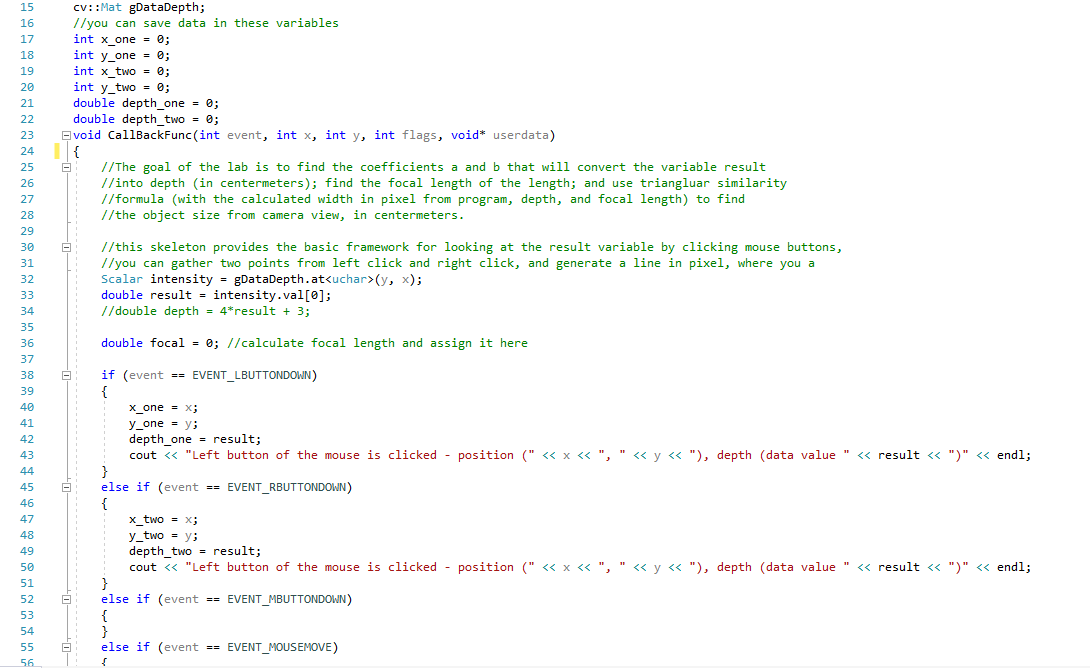
*C. Width Estimation*

Measure the width of an object, such as your phone, in centimeters. Record this value in each of the rows under “Width (cm)” in the table below. Place the object at five different depths away from the Kinect, and for each depth, record the pixel width of the object using the Kinect. Using the focal length formula we derived in class, assuming the object depth << image depth. Note that we measure the pixel width in pixels so the focal length in your formula is in fact a scaling factor you use later to convert pixel widths to actual object widths in cm. Find the focal length of the Kinect **in pixels** using the new focal length formula.

|  |  |  |  |
| --- | --- | --- | --- |
| Width (cm) | Depth (cm) | Pixel Width (pixel) | Focal Length (pixel) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Question 4:** What is the average focal length of Kinect in pixels?

If you look at the events in CallBackFunc below, we have assigned some variables to capture data of the first and second point you click on. To distinguish two points, you will left click on the first point and right click on the second point, the coordinates of which will be recorded in if statement blocks that handle events respectively. Similarly, EVENT\_MBUTTONDOWN handles the middle-click event.



**Question 5:** In the if statement block that handles MBUTTONDOWN, write a conversion equation that outputs , the distance between two points in cm, given relatively close depth. *Attach the code you wrote and a screenshot of it working.*

Hint: Use the focal length and the new triangle similarity formula you derived. You can keep the assumption of object distance and image distance we made earlier in this lab.

**Question 6:** In the if statement block that handles MBUTTONDOWN, rewrite the conversion equation that outputs , the distance between two points in cm, given **NOT** relatively close depth. *Attach the code you wrote and a screenshot of it working.*

Hint: Modify your equation in question 5 to account for two different depth values.

* 1. *Error Analysis*

**Question 7:** How well did your distance finder perform? What’s the relative error in percentage? Think of a couple ways to improve the result!

The raw Kinect depth value you recorded in the first table has in fact been processed through our code in the main function. Observe the code fragment below and look up what convertTo function does.



**Question 8:** What is the scale that converts the raw depth values from the frames of the video stream to the returned raw depth value we assign to the variable result? Hint: figure out what line 139 does and output iMaxDepth to get its value.

**Bonus Question:** What is the lowest nonzero raw depth value returned by the frames of the video stream? What is the unit of it? The OpenNI Sdk does not provide a way to configure depth stream to get a depth value smaller than this cutoff. Any depth values below the cutoff would return 0. Explain the reason of this setting. Hint: Explain a phenomenon you noticed when you placed you hands too close to the Kinect camera and justify why that would happen if possible.