Structured Light 3D Imaging Procedure Documentation

Environment Setup

1. Download and install the following programs
    a. https://docs.anaconda.com/anaconda/install/windows/
    b. https://www.meshlab.net/
    c. Canon EOS Webcam Utility Software - Canon Central and North Africa (canon-cna.com)
    d. https://code.visualstudio.com/download
2. Create an environment in the anaconda prompt. Make sure to replace <env-name> with your preferred environment name.
    a. conda create -n <env-name>
3. Run these commands in your anaconda environment to install the required libraries to run the code.
    a. conda install python=3.11
    b. conda install numpy
    c. pip install opencv-contrib-python==4.9.0.80
    d. pip install rawpy
    e. pip install open3d
4. We need to install openni2 separately. Download this file and once downloaded you can install primesense.
    a. https://structure.io/openni/
    b. pip install primesense
5. Next, we need to setup opencv for c++. We will do this using visual studio. Follow the guide from the link below. Follow the video from 3:50 – 11:50
    a. Firstly, install Visual Studio and during the installation process make sure to select "Desktop development with C++."
    b. Then install opencv using this link: https://github.com/opencv/opencv/releases/download/4.5.0/opencv-4.5.0-vc14_vc15.exe
    c. Next run the exe file and extract them into your file of choice.
    d. Then press the windows key and open "Edit the system environment variables," then click on the "Environment Variables" button.
    e. Then in the User Variables section double click on path and click new. Here you will paste the file route to the bin folder. Here is an example below
        i. C:\opencv\build\x64\vc15\bin
    f. After this restart your computer
    g. Create a new C++ project on visual studio and set the platform target to x64

h. Select your project on the right in the solution explorer. Right click it and select properties. Then select “VC++ Directories.” In “Include Directories” we can paste C:\opencv\build\include (These changes depending where you extracted the opencv folder. Then in Library directories paste this directory: C:\opencv\build\x64\vc15\lib
i. After this select the Linker drop down and select Input. Here edit the “Additional Dependencies,” and paste this: opencv_world450d.lib.
    i. NOTICE: the name of this file changes depending on what version of opencv you installed.
j. Now we can build and run the C++ file.
k. Credit to: https://www.youtube.com/watch?v=2FYm3GOonhk

Running the Code

1. Open the code through VSCode and then promptly exit out. Afterwards, go to your anaconda environment and type in code to enter VSCode. This ensures our environment is connected to our project.
    a. All the libraries should be installed and ready to use.
2. First run the Python file “GenerateArucoAndCharuco,” which is under the “CameraCalibration” folder.
3. After running you should have a file called charucoBoard.png. Print this out as this will be the board used for calibration.
    a. Make sure to place the board on a flat surface as any wrinkles or gaps can affect camera calibration.
4. Next, run the file “produceAllGrayImages.” This will create the structured light that we will be projecting onto our charuco board.
5. Then, run the file “CamTest” to check if your camera is properly connected to your computer.
    a. This also checks that you have properly installed the webcam software for your camera. (In our case the Canon Rebel T5i)
6. Run “OpenCVCamTest” to see if you are using the correct camera and what is seen by the camera shows up on your screen.
7. On your camera make sure the ISO value is not set to “AUTO” and set it to a specific value. For example, we set it to 1600. Also, make sure auto focus is turned off.
    a. This is done to keep our calibration photos consistent.
8. Now it’s time to create our set up. In our setup we place the projector directly in front of the board and the camera next to it facing the board. Furthermore, the Kinect is placed right below the camera.
    a. Insert Picture

9. Once the setup is prepared make sure most of the camera's field of view is on the charuco board.
10. Next, create a folder called "charucotest," in the captures folder. This will serve as our folder holding our photos. (You can call this folder whatever you want, however, you must change the name of the file the code calls later.)
11. Once this is complete make sure you set the projector as a second screen in your windows setting.
12. Next, place a linear polarizer on your camera. This will greatly reduce any potential glare that will show on the charuco board when the projector is on.
13. Now we can begin the picture taking process. Head to "CaptureCode" and run it.
    a. While it's running it will ask you to enter the name of the folder you want the pictures in (charucotest in our case).
    b. Then a new window will show up. Simply just drag it to the projector window and press enter. This will start the photo taking process.
    c. Once the photos are taken, end the code and move on to the next step.
14. Copy the folder where all the photos just taken are and paste them into your c++ project folder.
    a. For example, my file route looked like this: C:\Users\tree0\source\repos\GrayImageDecoder\GrayImageDecoder
    b. Yours may vary depending on where you installed Visual Studio.
15. Once this is complete, run the code, "GrayImageDecoder" in Visual Studio and you should see some tiff files in the same place where you pasted your folder of photos.
16. Next, in the captures folder create a folder called "Calib3." In this folder create a folder called "c_0." If more photosets are taken later on create another folder "c_1." (Simply counting up)
    a. The amount needed depends on the accuracy of our camera calibration matrix. For example, we only needed six sets of photos, but this may vary.
17. Once this is complete paste the .tiff files and photos taken into the "c_#" folder that you are currently on.
18. Repeat steps 13 – 17, while incrementing through each labeled folder in "Calib3."
19. To check your photos are accurate head to "main" in the "CameraCalibration" drop down and run it.
    a. If you receive an error about the number of corners or ID's being scanned, you need to retake that set of photos as not every corner and ID was found.
    b. Once this code can run without any issues a .npz file will appear in the "camera_calibration_out" folder called "calculated_cams_matrix."
20. To see the values in this file, run the "CameraCalibration\Calibration photos\npzReader.py" python file.
    a. Before running this make sure you change your directory to this using the prompt at the bottom in VSCode.

b. Change your directory using this command:
   cd C:\Coding_Projects\camera_calibration_out
c. The important thing to check is the two focal length values are about the same and that the return value (the first value printed) is under one. We want this number to be as close to zero as possible.
d. Although your return value can be close to zero after one set of photos, I recommend taking at least 5 sets of photos. However, the main goal is still to have a return value close to zero as this directly correlates to the quality of our point clouds.

Point Cloud Generation

1. Firstly, choose any object that you want to scan and place it in front of the charuco board.
2. Once you have this object set up and ready to go repeat steps 13 – 17. However, create a new folder in captures representing the item you are taking photos of and place the same incremented folders we had in "Calib3." This is where we will place all the photos, including the tiff files.
3. When the photos are set up head to the "main" file under the "ReprojectImage" drop down. Run this and you get a bunch of files found in the folder you made for the object images.
4. Find the filtered point cloud .ply file and open it in MeshLab. Here you can see your point cloud.

References

1. Sjnarmstrong. (n.d.). *SJNARMSTRONG/Gray-code-structured-light: Reconstructs a 3D scene with the use of a projector and a camera*. GitHub. https://github.com/sjnarmstrong/gray-code-structured-light
2. YouTube. (2020, December 13). *Learn OPENCV C++ in 4 hours | including 3X projects | computer vision*. YouTube. https://www.youtube.com/watch?v=2FYm3GOonhk