

TEEP Internship Final Report

Name: Aryan Rajesh

Internship Duration: July 2024 – September 2024

Project: Smart Tomato Harvesting Robot

Professor: Prof. Ping Lang Yen.

Mentor: Shaq

Team: Computer Vision

Internship Institution: National Taiwan University (NTU)

Home University: Presidency University, Bangalore, India

Introduction:

I am a research Intern at the RMML Lab under the Tomato Harvesting Project (Computer Vision) in the Bio-Mechatronics and Agricultural Department. I was selected for this program by Prof. Ping Lang-yen through the TEEP Program. I am immensely grateful to him for providing me with this excellent opportunity to learn so much about computer vision techniques, Neural Radiance fields and 3D reconstruction algorithms. I have gained many valuable skills.

So My Goal was to Generate the point Cloud of the whole Tomato Cluster and Process it. I was also instructed by my Mentors (Wei-Hsuan and Shaq) to Perform Gaussian Splatting and 3d Reconstruction of the Artificial Tomato Cluster. Its applications are not currently being directly implemented considering the current status of the project in future as the project advances it will surely be Useful.

Now, I will declare what work has been done and what needs to be done in the future.

Work Done Month Wise:

JULY:

My work Started with the idea that instead of Directly Obtaining the Point Cloud Why don't we just directly Extract the Point Cloud from the already available vast amount of 2d Images? I felt it was a unique Idea.

So I implemented, The Open AI's open-source Deep-Learning Point Cloud Extractor Model[1] From 2D Images. The model name is Point E by Open Ai....

I did not look further into this because I was told that this was not what the project demanded but in my opinion in future this DL- Model can be Really helpful. I gave a presentation on this topic on the 8th of July/24.

Furthermore To tackle the problem of the Image quality from which I am going to train the Open AI model I also created a Pipe-Line to first check whether the Depth of the image can be estimated before training the point E model to avoid any Bad Quality of Point Cloud Generation from the Open AI Model. This 2d Image Depth Estimation was done by Intel's Midas Model[2]

The Below Result Pictures are For your Reference:

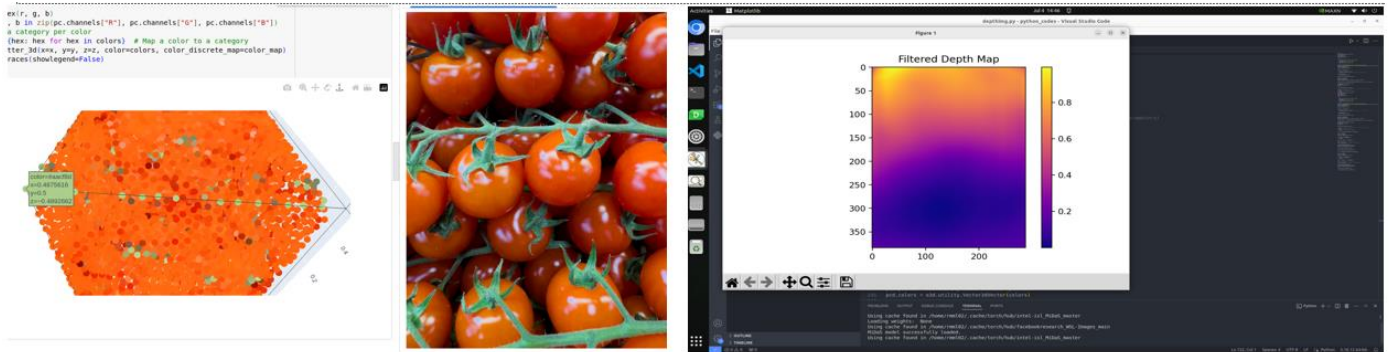


Figure 1: As we can see the Green line represents the Green Stem of the 2D image in the Tomato cluster (Only a Cross-Sectional Image) and the Filtered Depth map Created with Intel's Midas Model

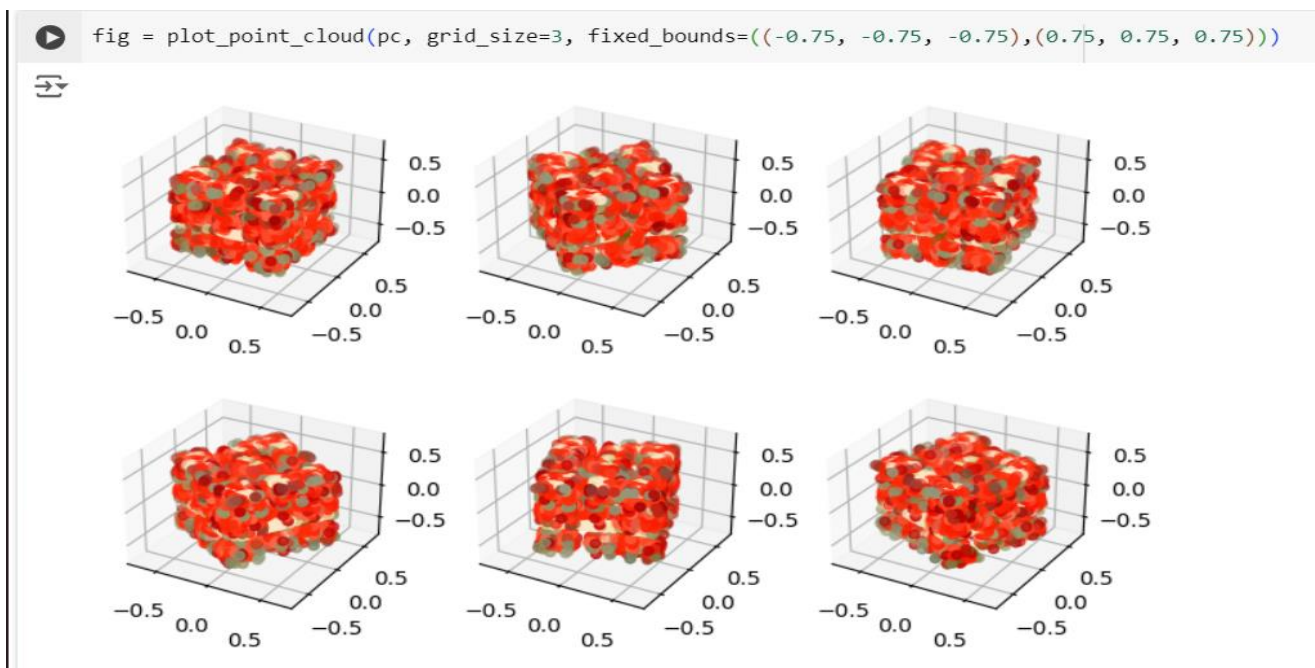


Figure 2: Small Cherry Tomatoes with green pedicle heads representation with different Angles, All these details are only from 1 or 2 2D images.

By mid-July I was Introduced To a new point Cloud Processing Software called *Cloud Compare* [3], I spent around a week learning and using this Tool To Process Captured Point Clouds. Then I closely Learnt about the ICP Algorithm and the Transformation Matrix of a Camera.

Now, By my own Developed Code, I captured a Point Cloud of a Tomato Cluster in Mesh Format but it was not clean so I added filters and optimised the Code further for faster Capture The Results are as Below Dated 8th July Presentation.

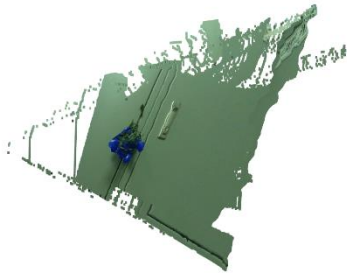


Figure 3: Before Raw Point Cloud With No proper Colour and missing Points



Figure 4: After Implementing My Code With the Colour Correction and Filling up the Missing Points

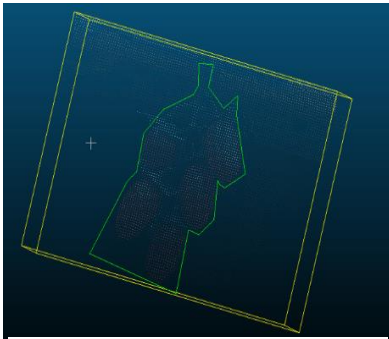


Figure 5 : Segmentation and cleaning. The Green Bounding Line shows the ROI(Kindly zoom in to see the Sparse Tomato point Cloud).

But this was the only mesh Format which Could be Used for 6D pose Estimation but it's a whole Cluster and not a single Tomato.

Also Performing the Cleaning Background of the Point Cloud.

Moving ahead I started Capturing 3D data directly in Raw Point Cloud Format instead of Mesh Format (Ply format). Below Figure 6, the Image can be referenced and compared to Figure 2.

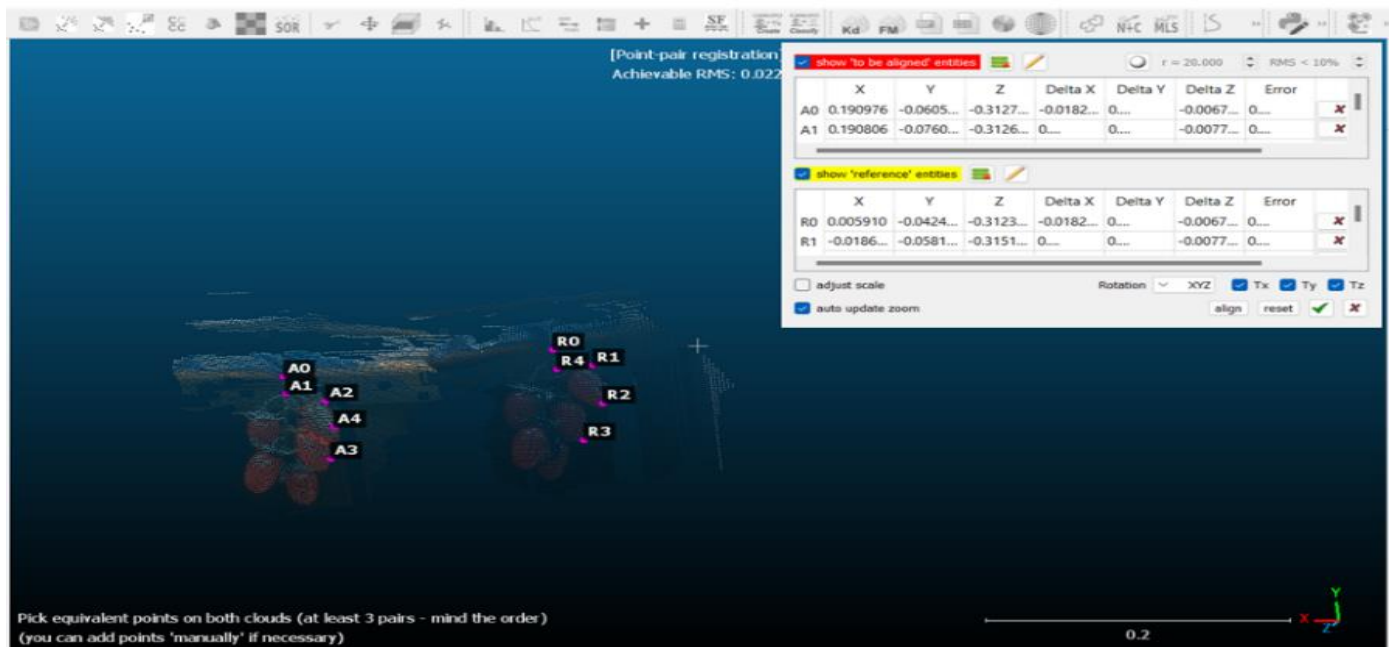


Figure 6: This shows that the 2 Raw Point Clouds are Captured and Point-Pair Registration has taken Place From one Point Cloud to another

By the End of July, I was Applying the ICP algorithm on 2 Point Clouds Which Takes in The Transformation Matrix as Input selects the Points for Coarse Alignment and then proceeds to the Fine Tuning Alignment. We Have to zoom in to see the Point Cloud.

In Fig 7. The left side is the Transformation Matrix of FIG.8 The Point Cloud and the Right side depict the Points in the Point Cloud dated 15th/July/2024 Presentation.

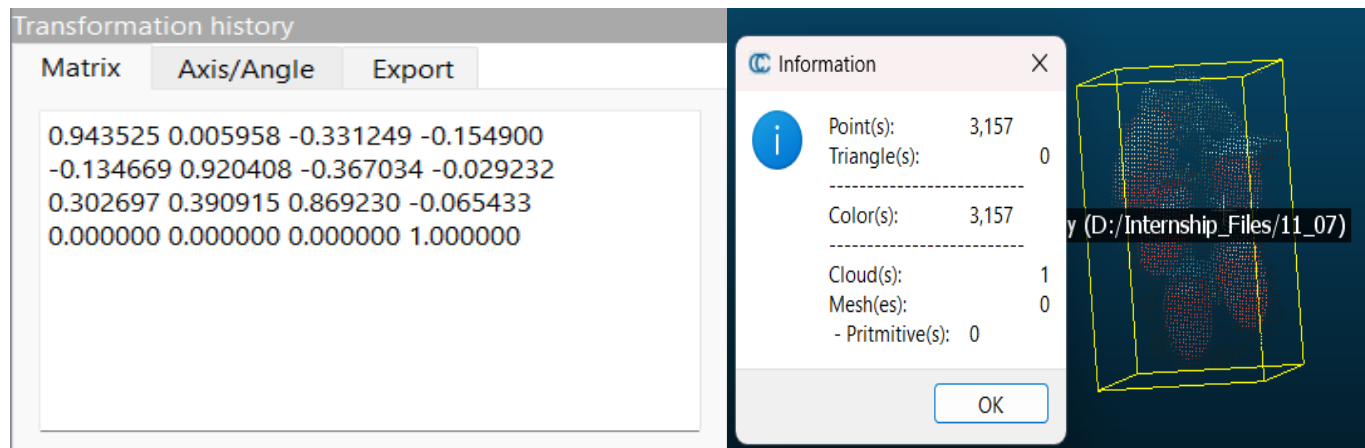


Figure 7: Transformation Matrix and Information of the obtained Point Cloud which I used in the ICP Algorithm for Point-Pair Registration.

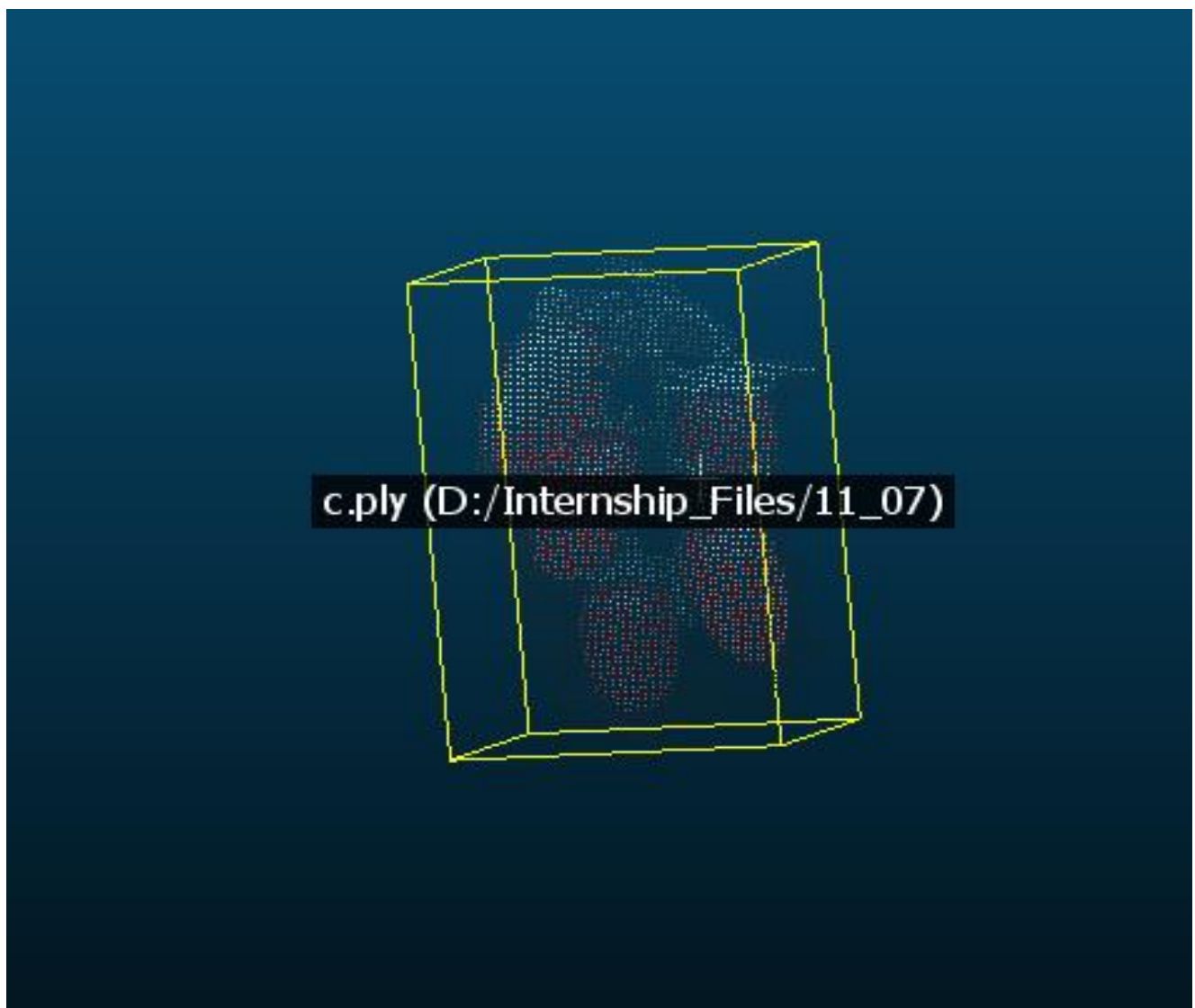


Figure 8 : Point Cloud Captured! Of the tomato cluster; I also feel that the quality is quite acceptable (Raw Point Cloud Format/ply)

August:

I was told that we have to perform pose estimation for the robotic arm to orient itself to the tomato cluster and also perform 3D reconstruction. So for This, I performed LoFtr Analysis taking Reference from the Related Reference Papers [4],[5],[6].

LoFTR is an attention-based DL Learning method designed to match points in two images across large viewpoint differences which can be used for Pose Estimation and 3D Reconstruction from 2D images.

The Reasons of Conducting Loftr Analysis are Listed Below:

- **Detector-Free Matching:** This Feature Based Matching eliminates the need for Separate KeyPoint Detection and Algorithm
- **Improved Tracking Robustness:** LoFTR's dense matching reduces tracking failures in VO pipelines.
- **Enhanced Pose Accuracy and Estimation:** It Provides more precise feature correspondences leading to better pose estimation and Object Poses.
- **Real-Time Performance:** Optimizes LoFTR's implementation to achieve low-latency processing suitable for real-time applications.

The images below show the Loftr analysis in real-time and it can be observed that with Each Iteration of the code, the 2d Matching Takes between Different Images and at the Start Barely a Few KeyPoint Features are Matched but at the end of the Process as we can see there are is Significant Increase of Feature Matching.

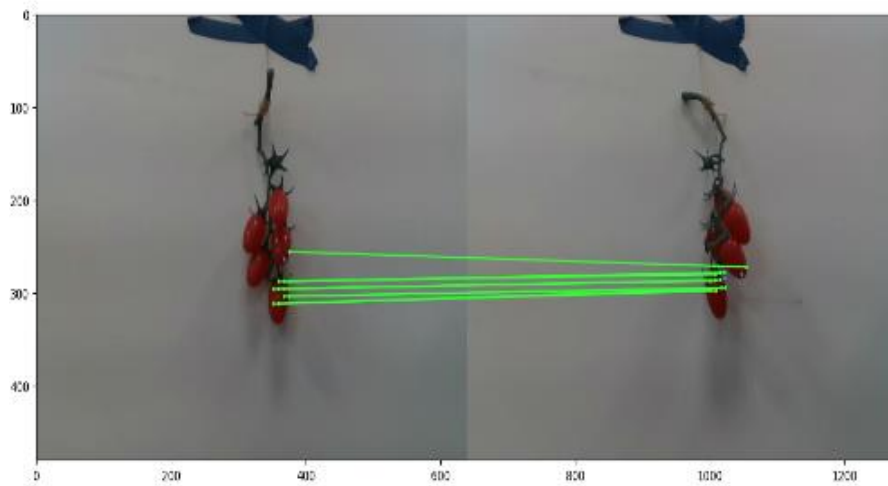
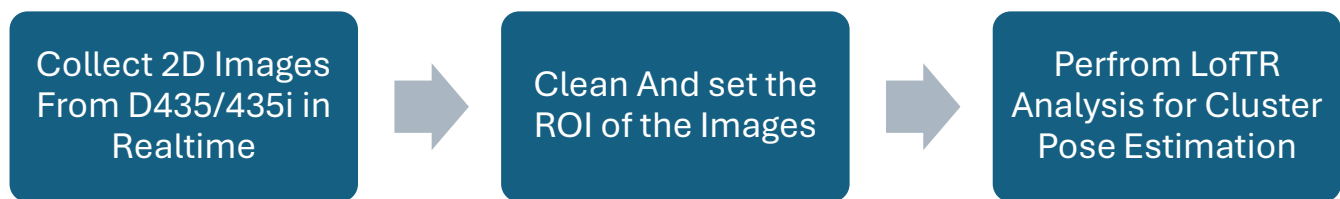


Figure 9: At the Start, only a Few Features were Matched(Accuracy 19%)

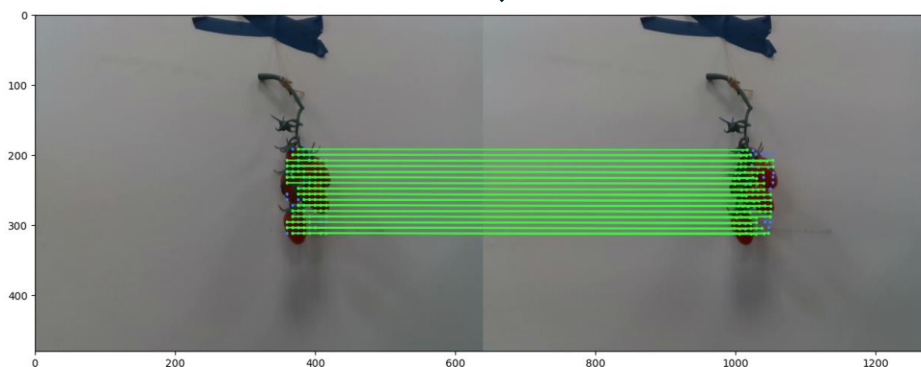


Figure 10: All The Keypoint Feature Matching is Done from 2D images (Accuracy 89%)

Additionally, I also Did Camera Calibration for the keypoint Detection to Work properly This Was done by taking the Point Cloud of the keypoint Frame and then Performing the Calibration Please Observe the Green Dots below as First It's Recognizing Random Black Spots after Calibration it can Detect the Keypoint. Now the Purpose of this is that Point Cloud of the Tomato can also be Obtained From this method as Instructed in this GitHub Repository.... Before Calibration and After Calibration.

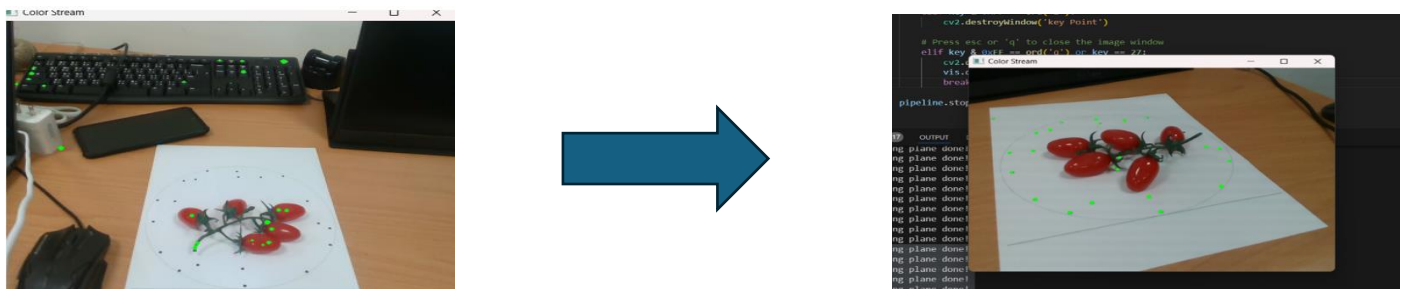


Figure 11:

Then I continued to Extract Point Cloud from the Commercial Software (RecFusion Pro)[7]. I Tried Multiple Ways to Use this Software most Efficiently. I ended up Using a MULTI-CAM Reconstruction Method But these cameras had to be calibrated Sequentially so I used the Arcuo Marker 200mm And the ID was 200. I discontinued this Approach because the software was Commercial, but I do think that it performs well in Mesh 3D Reconstructions.

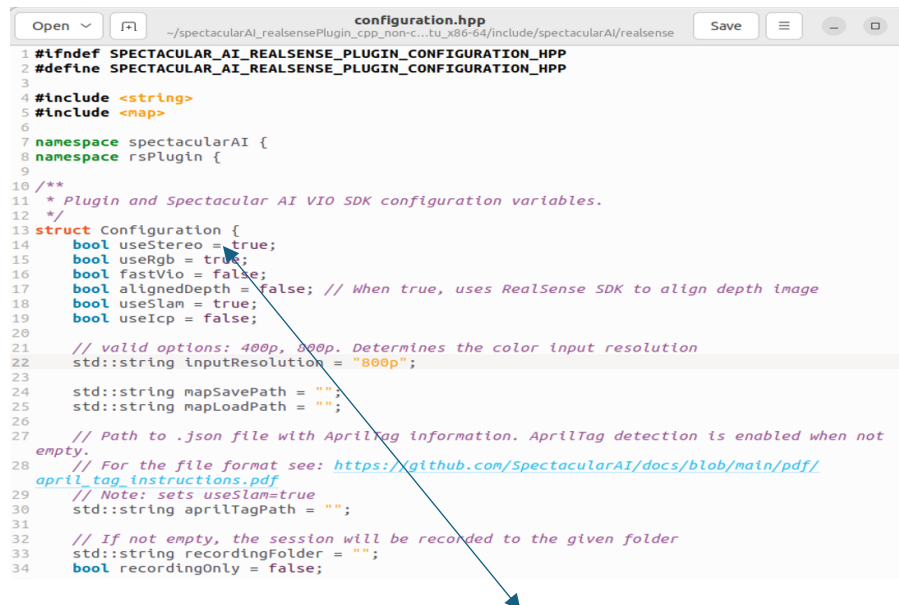


Figure 12 : As we can see 2 Point Clouds are Seen in Figure 13: Only 1 Merged Point Cloud is shown because of Calibration

September:

I was instructed by my Senior Wei-Hsuan and Mentor Shaq to Perform Gaussian Splatting (3DGS)[8] on the Tomato Cluster and was Given Multiple Research Papers To review, Study and then Reconstruct the Tomato Splat via the Spaltfacto Model.

But for this, I first needed a good 180 Degree Point Cloud of the Tomato So I conducted V-Slam via an open-source Library called Spectacular Ai Realsense[9] this plugin works on Linux x64_x84 & Windows Platforms but there are problems running it on the Arm Architecture Devices like the Orin or the nano (My suggestion if in future anybody who wants to run this program can Virtualize this program in Orin/Nano with a help of an emulator it can be any emulator For ex. Box_64x_86x and Qemu). Either way, this Was useful for me to extract the Sparse point cloud and RGB/depth images with real-time Camera poses with their Transformation Matrix and Perform The Gaussian splatting. This was especially helpful for me to use because we can easily edit the Source code of this and Choose our preferred operation mode based on our requirements for e.x. ICP, Slam, Resolution, Realsense Post Processing Filters etc.



```

1 #ifndef SPECTACULAR_AI_REALSENSE_PLUGIN_CONFIGURATION_HPP
2 #define SPECTACULAR_AI_REALSENSE_PLUGIN_CONFIGURATION_HPP
3
4 #include <string>
5 #include <map>
6
7 namespace spectacularAI {
8 namespace rsPlugin {
9
10 /**
11  * Plugin and Spectacular AI VIO SDK configuration variables.
12  */
13 struct Configuration {
14     bool useStereo = true;
15     bool useRgb = true;
16     bool fastVio = false;
17     bool alignedDepth = false; // When true, uses RealSense SDK to align depth image
18     bool useSlam = true;
19     bool useIcp = false;
20
21     // valid options: 400p, 800p. Determines the color input resolution
22     std::string inputResolution = "800p";
23
24     std::string mapSavePath = "";
25     std::string mapLoadPath = "";
26
27     // Path to .json file with AprilTag information. AprilTag detection is enabled when not
28     // empty.
29     // For the file format see: https://github.com/SpectacularAI/docs/blob/main/pdf/april\_tag\_instructions.pdf
30     // Note: sets useSlam=true
31     std::string aprilTagPath = "";
32
33     // If not empty, the session will be recorded to the given folder
34     std::string recordingFolder = "";
35     bool recordingOnly = false;
36 };
37
38 }
39 }

```

Figure 14: We can Choose our preferred mode of operations.

Moving ahead I Will demonstrate the Slam and Point cloud of the Tomato Cluster this was taken in 800p resolution it is a bit resource-extensive Process but it is accurate it works with the IMU sensor built Inside the D435i. **Note This plugin will not work with the d435 camera without the 'I' because the d435 lacks the imu sensor Module and cannot write the camera poses into the vio.json file because of lack of the imu sensor which is essential for reconstruction.

The specifications of my setup while performing the V-slam via Spectacular Api are as Follows:

- Lab Computer – Linux_64x_86x Architecture
- Ubuntu 22.04
- Rtx-3060 GPU
- Nvidia Cuda-Toolkit 11.8
- Nvidia – 535 Driver Package(*Important to Match with the 11.7/8 Cuda Version)

The figures below show the progress of the V-Slam from the start to the end. **In point Cloud format.

Please note that this is an approximately 220-degree Reconstruction of the artificial tomato cluster conducted in a lab with sufficient lighting. One Important Point to note is that the below images show the progress of the V-Slam where the camera is moved from the left side to the right side in a circular trajectory to capture the best view. The Orange Box Depicts the Position in which the Code will Capture 2 Images Simultaneously (1 RGB image and 1 Depth Image) The Orange Box Depicts the Position in which the Code will Capture 2 Images Simultaneously (1 RGB image and 1 Depth Image). The Red Box Depicts the Live Position of the Recording Device(Realsense D435i). The Blue Line depicts the Covered Trajectory/Path Covered by the camera. Please note That I have fast-forwarded the Video 2.5x Times for Presentation Purposes. In reality, it is much slower and should be recorded gently without aggressive shaking of the camera. The Red Box Depicts the Live Position of the Recording Device(Realsense D435i). The Blue Line depicts the Covered Trajectory/Path Covered by the camera.

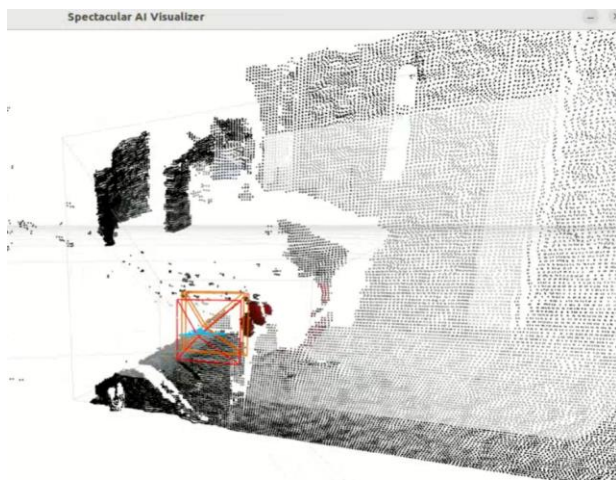
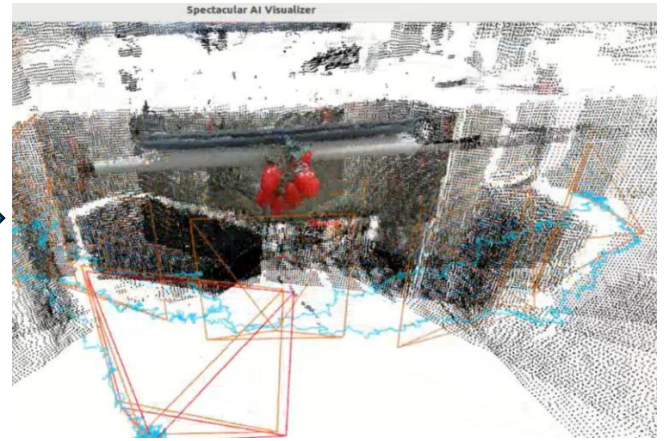


Figure 15: *Left Image:- The start of the Slam



*Right Image:- Completed Slam In point Cloud Format

So now we Perform Gaussian Splatting with the help of the data recorded by the Spectacular AI in .MKV format, for this, we first need to process the data collected by the plugin to train the 3DGS Model. Additionally, this plugin supports multiple Processing format outputs for training taichi 3D Splatting and Nvidia-Instant-NGP(Bounded/UNBounded). It extracts the camera pose, depth frames, Rgb Frames, and Calibration Json in any of the mentioned above-preferred formats. Also, the sai-cli command file (.py) can be changed and edited depending on the Output Processing needs easily (If required for some other Task).

Now that we have our data ready for our 3DGS model to train we just need one more thing and that is **NerfStudio[10]**. It's as simple as plug and play with nerfstudio! Nerfstudio provides a simple API that allows for a simplified end-to-end process of creating, training, and testing NeRFs. The library supports a **more interpretable implementation of NeRFs[11] by modularizing each component**.

So first I trained the Legacy Nerf-Studio Model (NerfFacto) and I did not get good results as it took approx. $\approx 02:30:00$ time to Train a simple model with below-average Resolution which I Was not satisfied with as we can see below I have Attached the Training Period and the Result also.

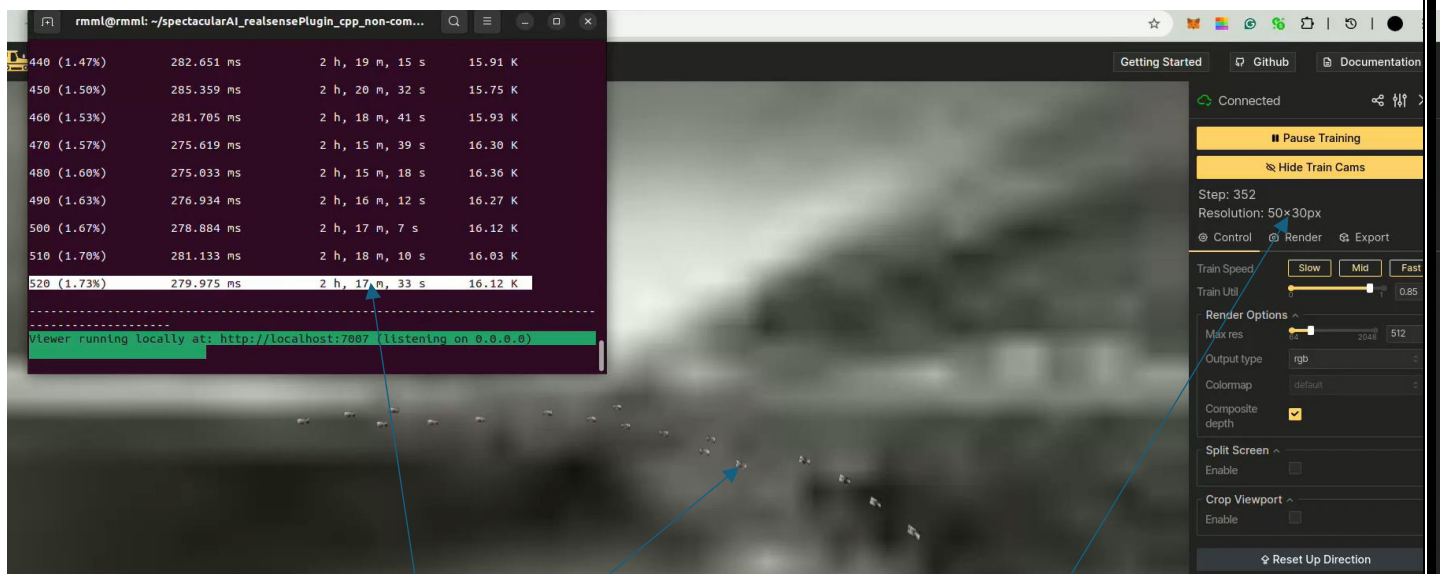


Figure 16: Please Observe the Training Time(Approx 2 hours 17 minutes) and also please Observe the Resolution(50*30) of this Nerfacto Model (Too much Time and Very Less Resolution). These boxes represent the Camera Poses and when the image was captured of the Cluster.

Now Please compare the Training time and Resolution of the Nerfacto and 3DGS model.

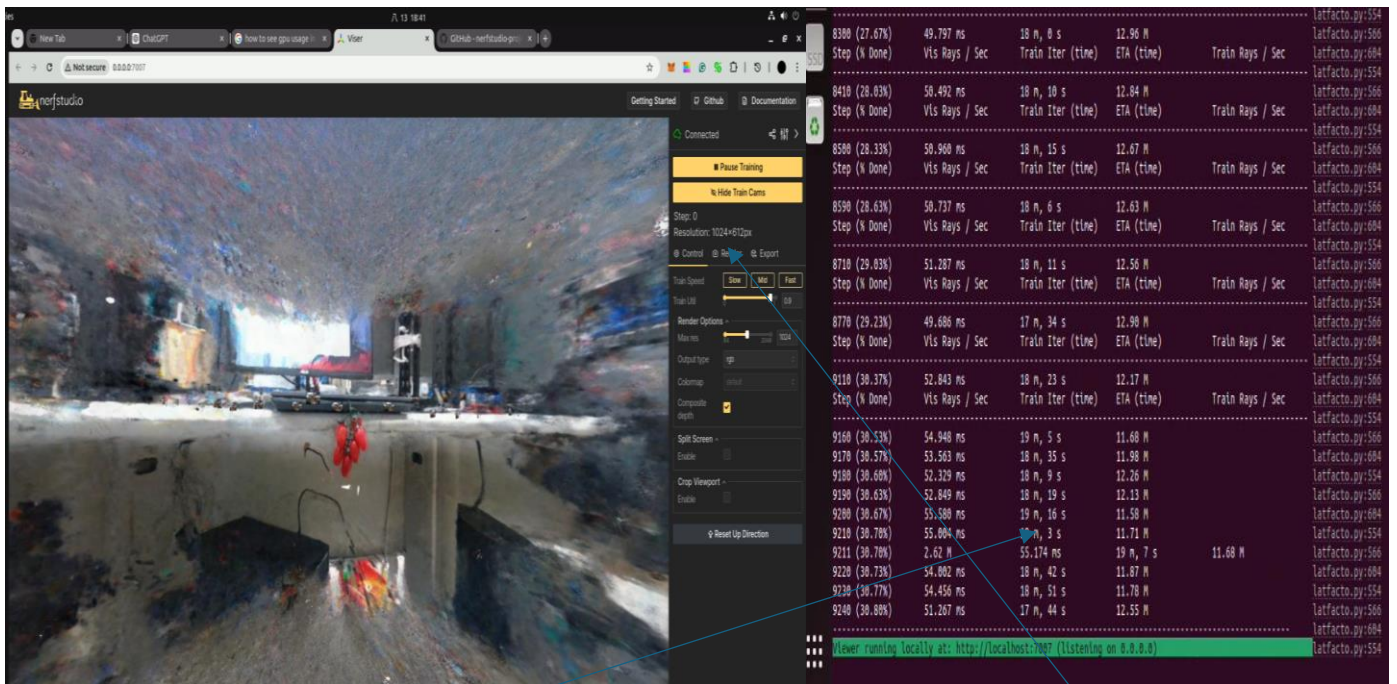


Figure 15: This is the Gaussian Splatting Model(Completed 30-35 FPS), As we can see the Quality and the **Resolution(1024*612)** and also the **time taken(Approx \approx 19 minutes)** To complete a Training

I am also Attaching the Resource Consumption Statistics of the Graphical unit, The Better the GPU lesser the Time taken to train the model but all GPUs will still generate the same quality and resolution of the model only the time taken to generate will vary.

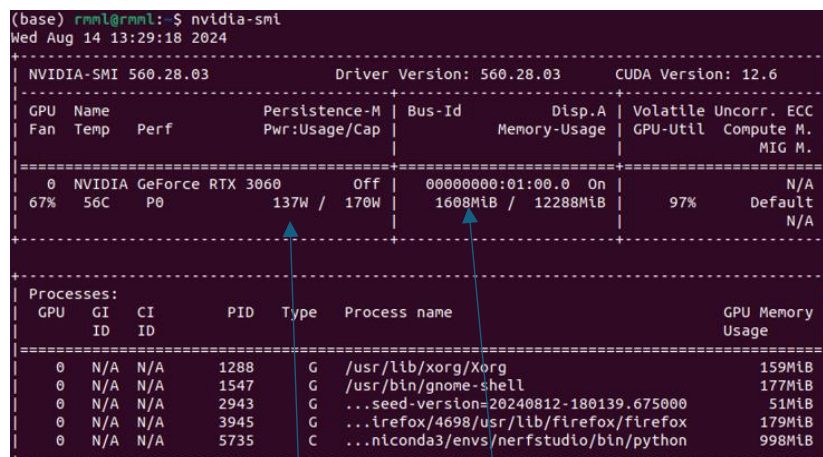
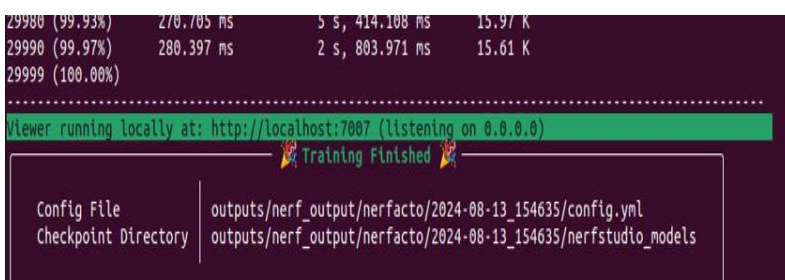


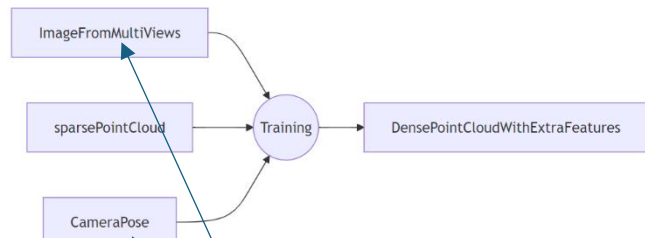
Figure 16: This shows the **Power(137watt)** and Memory usage of the while Performing 3D Reconstruction



The number 0029999 means that the model has completed these many Steps to complete the 3D reconstruction we can increase the number of steps in the core library but will take Considerably more time and resources. Ckpt stands for Checkpoint so next time someone loads the model the code will directly load from the 29999th Checkpoint.



Also, the Location of the Model I have Given Below includes the path to both The Checkpoint File and the Config.yml file which looks like this. Anybody can take it from the Lab PC and use it again to modify it. Gaussian Splatting is Done!



```

1 {
2   "aabb_scale": 16,
3   "auto_scale_poses_override": false,
4   "cx": 323.515,
5   "cy": 175.171,
6   "fl_x": 459.066,
7   "fl_y": 458.398,
8   "frames": [
9     {
10      "camera_angular_velocity": [
11        0.001265842903331676,
12        0.009302584668543725,
13        0.005619243293719566
14      ],
15      "camera_linear_velocity": [
16        0.05263555588050407,
17        -0.07044438397778442,
18        0.018279763173383805
19      ],
20      "depth_file_path": "./images/depth_00001.png",
21      "file_path": "./images/frame_00001.jpg",
22      "motion_blur_score": 21.618351406897226,
23      "transform_matrix": [
24        [
25          0.44455447037850104,
26          0.11445333761451326,
27          -0.8884096782314954,
28          0.07177044136973708
29        ]
30      ]
31    }
32  ]
33 }
  
```

My Mentor also told me To go to the Extra Mile and perform Taichi 3D Gaussian Splatting to improve the Point Cloud Quality which can be applied in other Ongoing Projects not only in the current ongoing Project. To give a Quick Explanation of the Taichi 3D Splatting this flowchart might help.

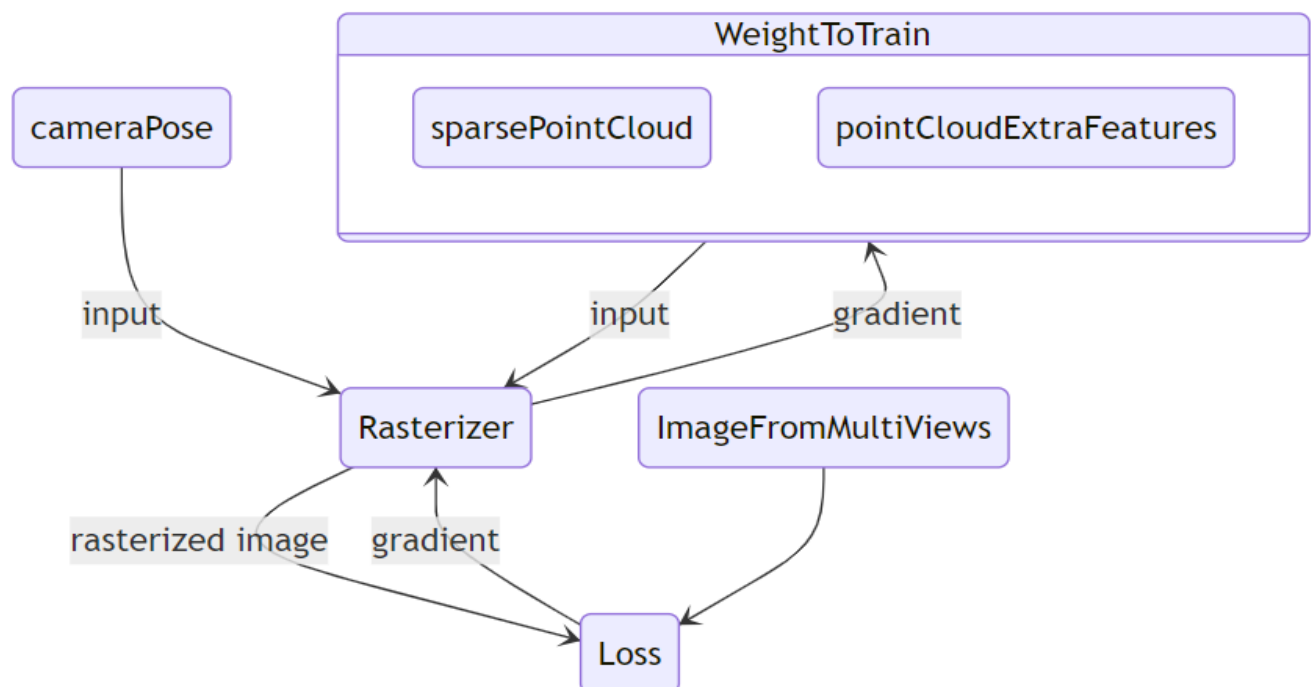
This file called “transforms.json” contains the transformation Matrix of all the Sequential Images I captured while recording it contains the Calibration Matrix of the Intel Realsense Camera. Also contains the Path to all the Depth and RGB Images for the Training. The command line to this for Taichi is given below.

The Inference➔ The algorithm takes the dense point cloud with extra features and any camera pose as input, and uses the same rasterizer to render the image from the camera pose.

“sai-cli process --format=taichi ... -input path - Output path”

If anybody wants to understand the Taichi 3D Explanation in brief this might help.

The training process works in the following way:



Though this was almost at the End of my Internship that’s why I couldn't refine the result that well and did not continue this path because I was told to focus on another Topic.

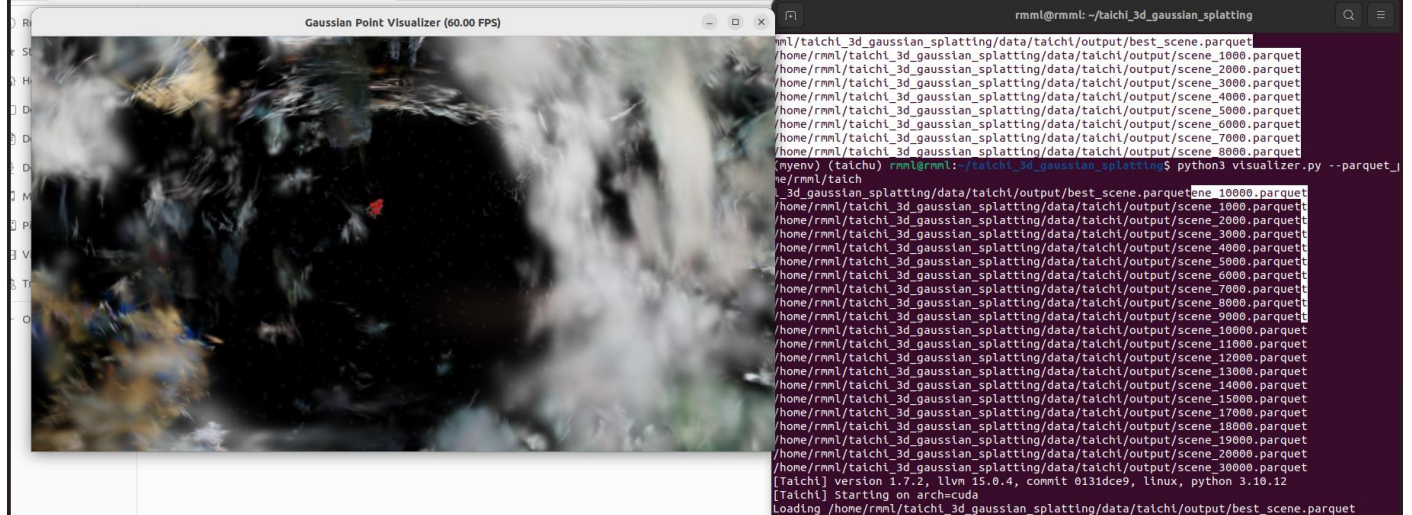


Figure 17: Visualization of Taichi Model at 60.00 FPS using the Visualizer which merges multiple Parquet Point Cloud files

Steps to Run the Visualizer:

Access the → `python3 visualizer.py --parquet_path_list <parquet_path_0> <parquet_path_1> ...`

The visualizer merges multiple point clouds and displays them in the same scene.

- Press 0 to select all point clouds(default state).
- Press 1 to 9 to select one of the point clouds.
- When all point clouds are selected, use "WASD=/" to move the camera, and use "QE" to rotate by the y-axis, or drag the mouse to do free rotation.
- When only one of the point clouds is selected, use "WASD=/" to move the object/scene, and use "QE" to rotate the object/scene by the y-axis, or drag the mouse to do free rotation by the centre of the object.

In the last week of my Internship, I was told to collect the Dataset of the Tomato Cluster so we had to scan every tomato piece to form a suitable Textured 3D Tomato Model for the 6D Pose Estimation. So tried to come up with a 2 Step Approach to Form a dataset but due to the time constraint I could only Manage the 1st Step:

- ▶ Scan And Record
- ▶ Perform Reconstruction of each tomato

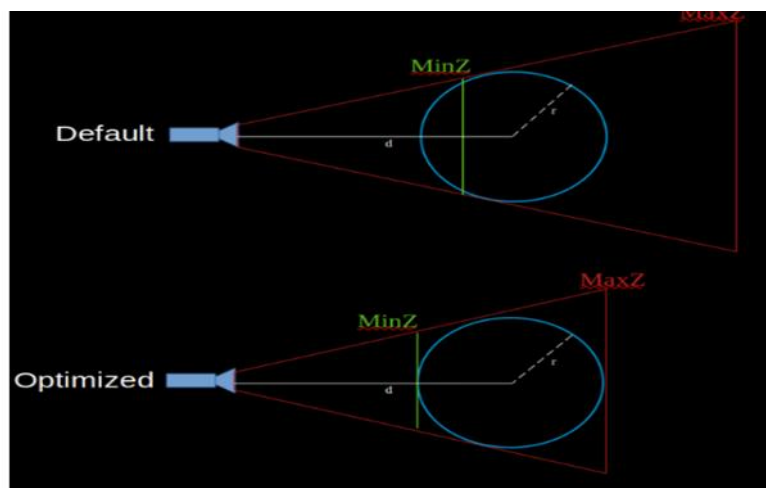


Figure 18: The Working Principle

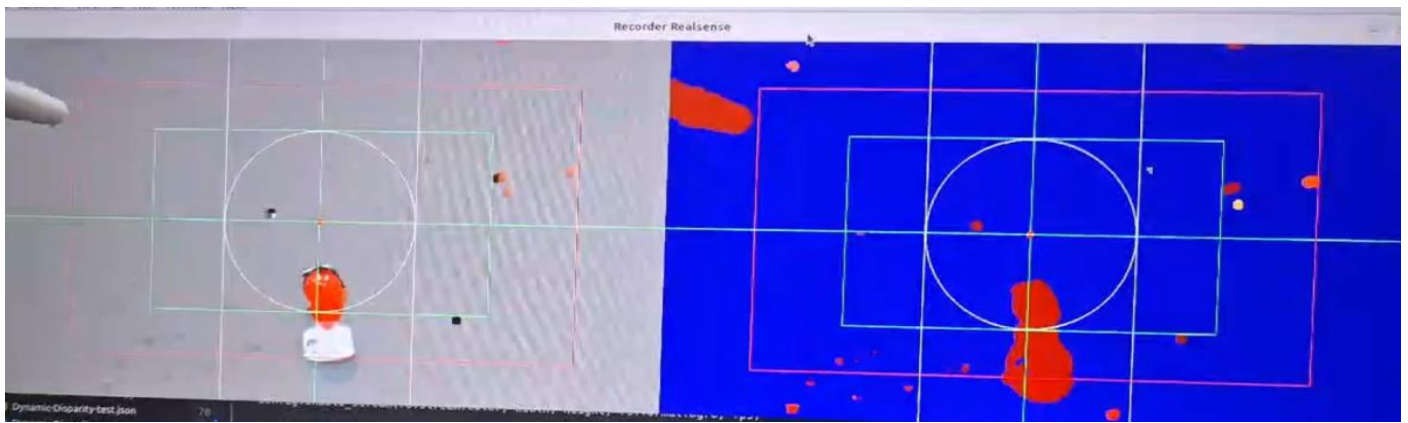


Figure 19: The left frame shows the RGB Feed and the Right Feed shows The Depth Feed

```

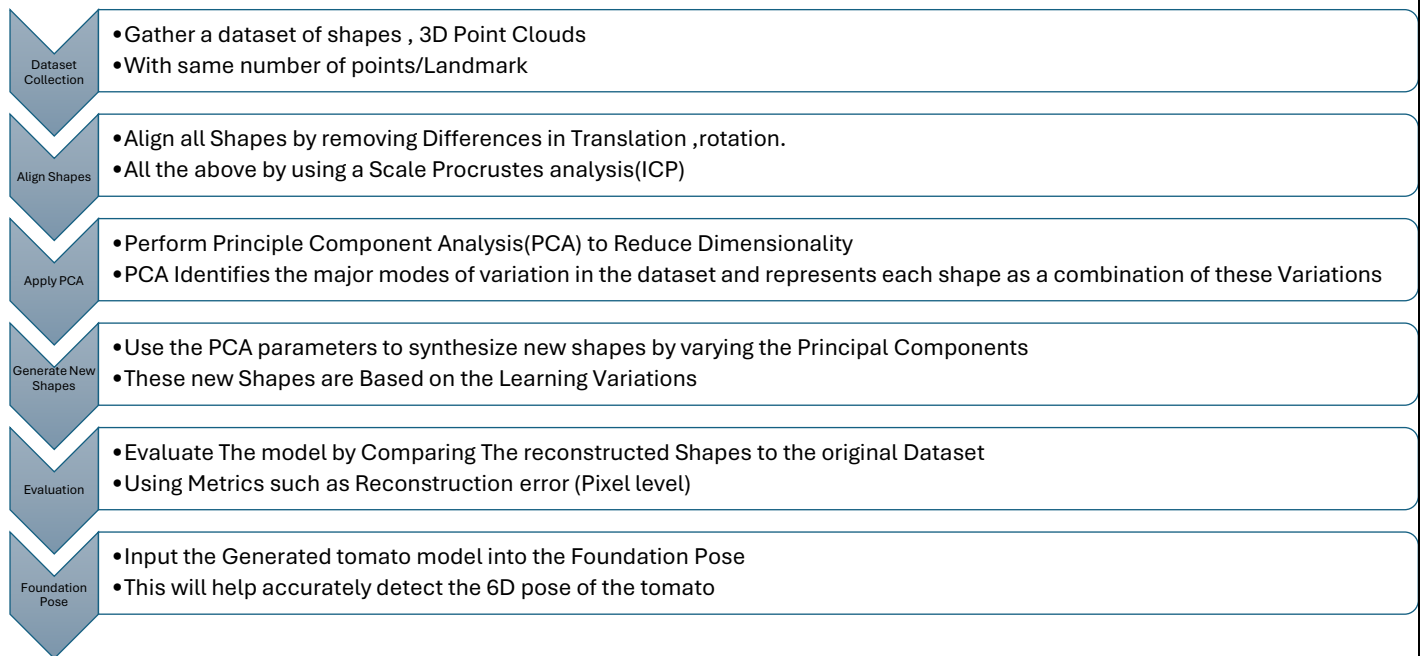
79
80 # Create folders
81 path_color = join("dataset", "color")
82 path_depth = join("dataset", "depth")
83 make_clean_folder(path_depth)
84 make_clean_folder(path_color)
85
86 # Create pipeline
87 pipeline = rs.pipeline()
88
89 # Configure streams
90 config = rs.config()
91 config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
92 config.enable_stream(rs.stream.color, 640, 480, rs.format.rgb8, 30)
93
94 # Import custom preset
95 jsonObj = json.load(open("/home/aryan/lffinal/3D-scan-and-reconstruction-realsense-d435i/MediumDensityPreset-custom.json"))
96 json_string = str(jsonObj).replace("'", '"')
97
98 # Resolution and FPS
99 width = int(jsonObj['viewer']['stream-width'])
100 height = int(jsonObj['viewer']['stream-height'])
101 fps = int(jsonObj['viewer']['stream-fps'])
102 print("W: ", width)
103 print("H: ", height)
104 print("FPS: ", fps)
105
106 Saved color + depth image 000497
107 Saved color + depth image 000498
108 Saved color + depth image 000499
109 Saved color + depth image 000500
110 Saved color + depth image 000501
111 Saved color + depth image 000502
112 Saved color + depth image 000503
113 Saved color + depth image 000504
114 Saved color + depth image 000505
115 ^ rotation 1
116 476 frames recorded
117 ariyan@ariyan-virtual-machine:~/lffinal$
  
```

Figure 20: 505 images are saved but the code automatically skips the Blurry Frames and is reduced to 476 Frames The Visualizer is automatically terminated after 1 Rotation.

I Still need time to work on the Reconstruction Algorithm. The above code scans the 360 Degree of the Tomato automatically detects the Rotation and Saves the Depth and RGB frame.

Future Work:

We need a Dataset of fresh Cherry Tomatoes for 6D Pose Estimation. And Develop a Statistical Shape Model Classifier Machine learning Algorithm to form a suitable shape model of the tomato to feed in the Foundation Pose Api[13].



References :

- [1] Nichol, A., & Dhariwal, P. (2022). *Point-E: A System for Generating 3D Point Clouds from Complex Prompts*.
- [2] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 1623-1637.
- [3] CloudCompare (Version 2.x) [Computer software].
- [4] Ding, C., Dai, Y., Feng, X., Zhou, Y., & Li, Q. (2023). Stereo vision SLAM-based 3D reconstruction on UAV development platforms. *Journal of Electronic Imaging*, 32(01).
<https://doi.org/10.1117/1.jei.32.1.013041>.
- [5] *PViT-6D: Overclocking Vision Transformers for 6D Pose Estimation with Confidence-Level Prediction and Pose Tokens*. (n.d.). <https://arxiv.org/html/2311.17504>
- [6] Adityamwagh. (n.d.). *GitHub - adityamwagh/3d-reconstruction-loftr: Pose estimation pipeline for 3D Reconstruction using LoFTR (Local Feature Transformer) detector free feature matcher*. GitHub.
<https://github.com/adityamwagh/3d-reconstruction-loftr>
- [7] RecFusion - 3d Reconstruction Software. (2023, January 16). *Download RecFusion | RecFusion - 3d Reconstruction Software*. <https://www.refusion.net/download-recfusion/>
- [8] *Gaussian Splatting & NeRFs — spectacularAI documentation*. (n.d.).
<https://spectacularai.github.io/docs/sdk/tools/nerf.html>
- [9] Graphdeco-Inria. (n.d.). *GitHub - graphdeco-inria/gaussian-splatting: Original reference implementation of “3D Gaussian Splatting for Real-Time Radiance Field Rendering.”* GitHub.
<https://github.com/graphdeco-inria/gaussian-splatting>

- [10] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., & Kanazawa, A. (2023). Nerfstudio: A Modular Framework for Neural Radiance Field Development. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2302.04264>
- [11] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2003.08934>
- [12] Wanmeihuali. (n.d.). *GitHub - wanmeihuali/taichi_3d_gaussian_splatting: An unofficial implementation of paper 3D Gaussian Splatting for Real-Time Radiance Field Rendering by taichi lang*. GitHub. https://github.com/wanmeihuali/taichi_3d_gaussian_splatting
- [13] Wen, B., Yang, W., Kautz, J., & Birchfield, S. (2023). FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2312.08344>