Multi-Camera 3D Fusion with BlenDR

Joon Ha Kim

20180897

Contents

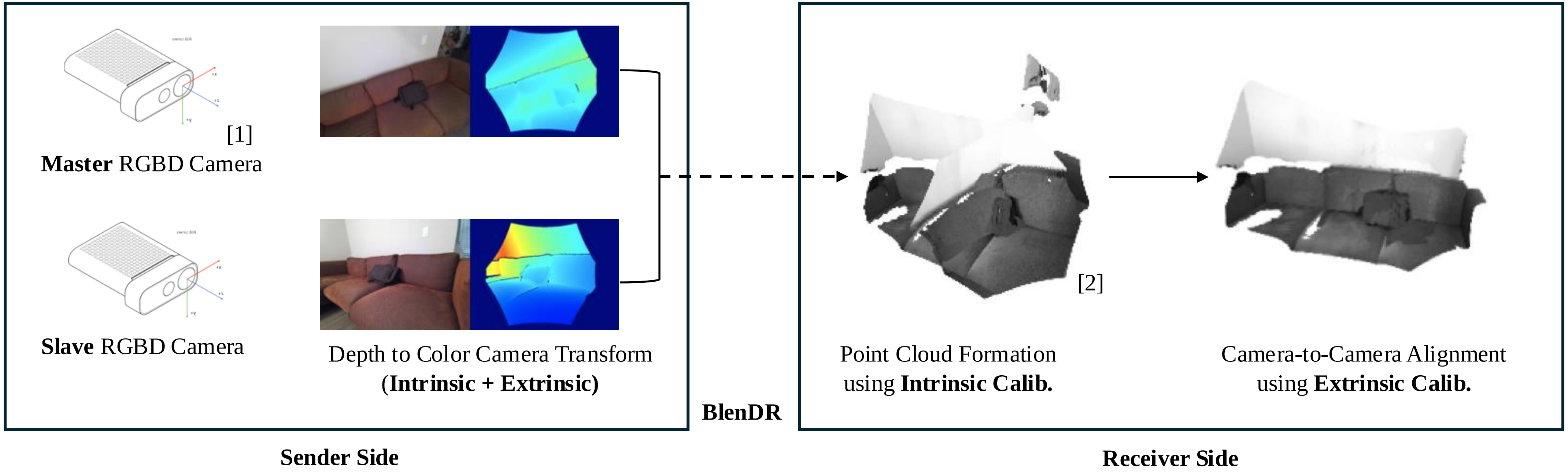
- Revisit Project Progress
- Multi-Fusion BlenDR System Design
- Progress Update
- Remaining Work

Project Progress Revisited

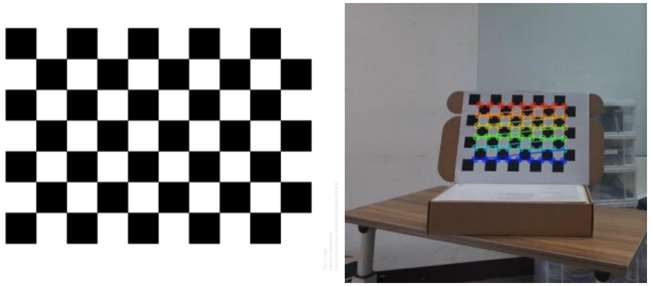
Progress from March to April 2024

Project Details

- **Goal:** Fuse multi-view point clouds to transmit (using BlenDR) a dense point cloud for improved spatial and temporal consistency
- **Key Terms:** Master/Slave Camera, Intrinsic/Extrinsic Calibration

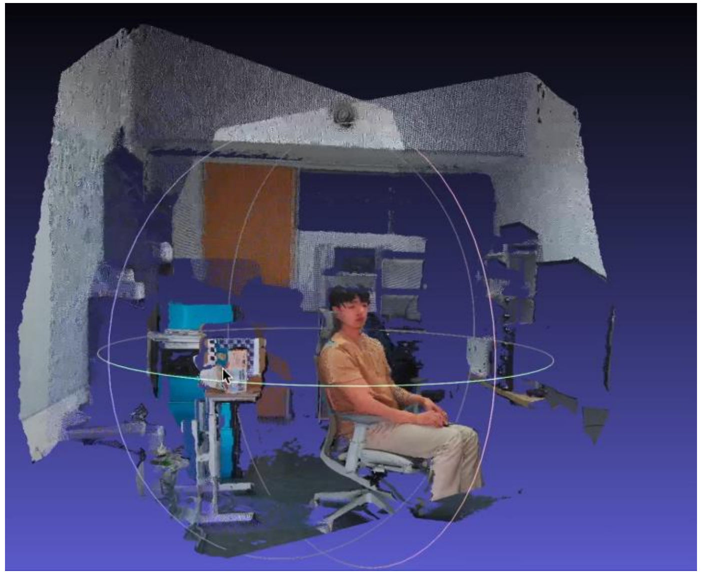


Summary of Fusion Process



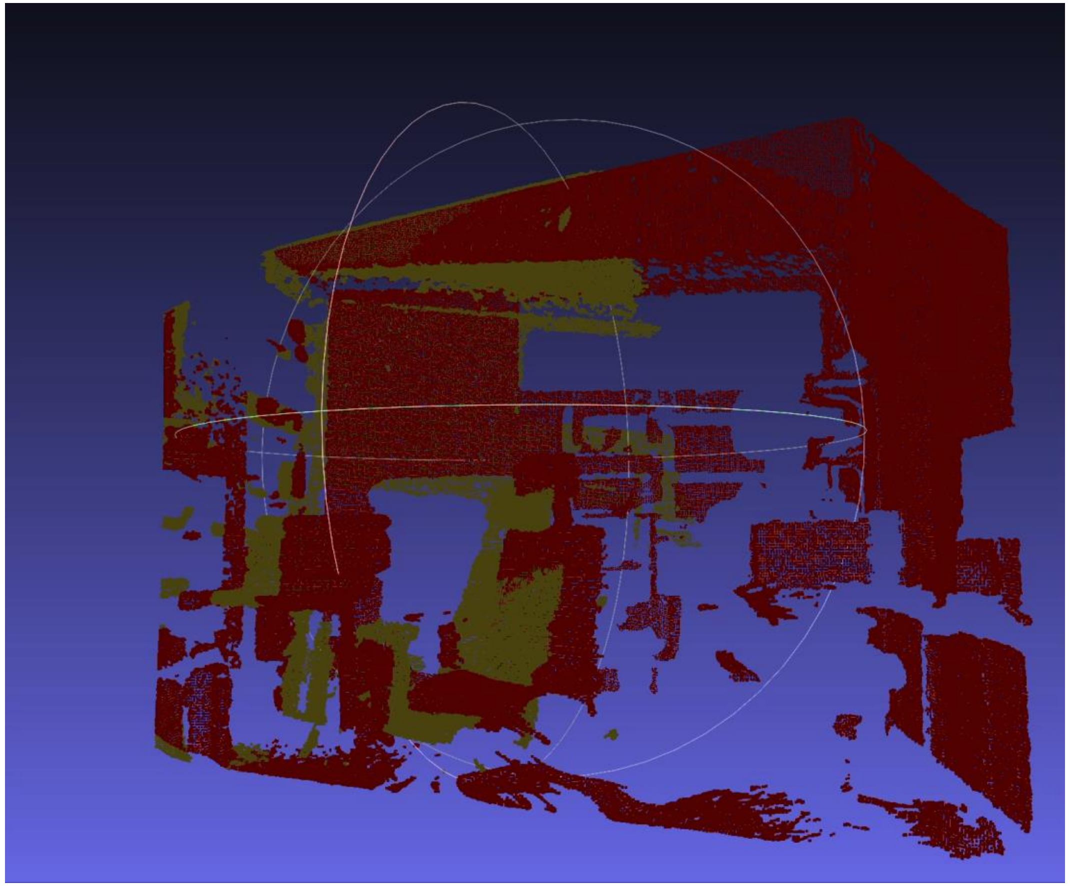
Calibration using Checkboard (slave and master)

1. Retrieve **external calibration** data through **checkerboard**
2. Retrieve **internal calibration** to **make point cloud** from each view
3. Use **external calibration** to transform the slave point cloud **(Stereo Calibration)**
4. Use **ICP Algorithm** (in Appendix) to create a more accurate fusion of two pointclouds

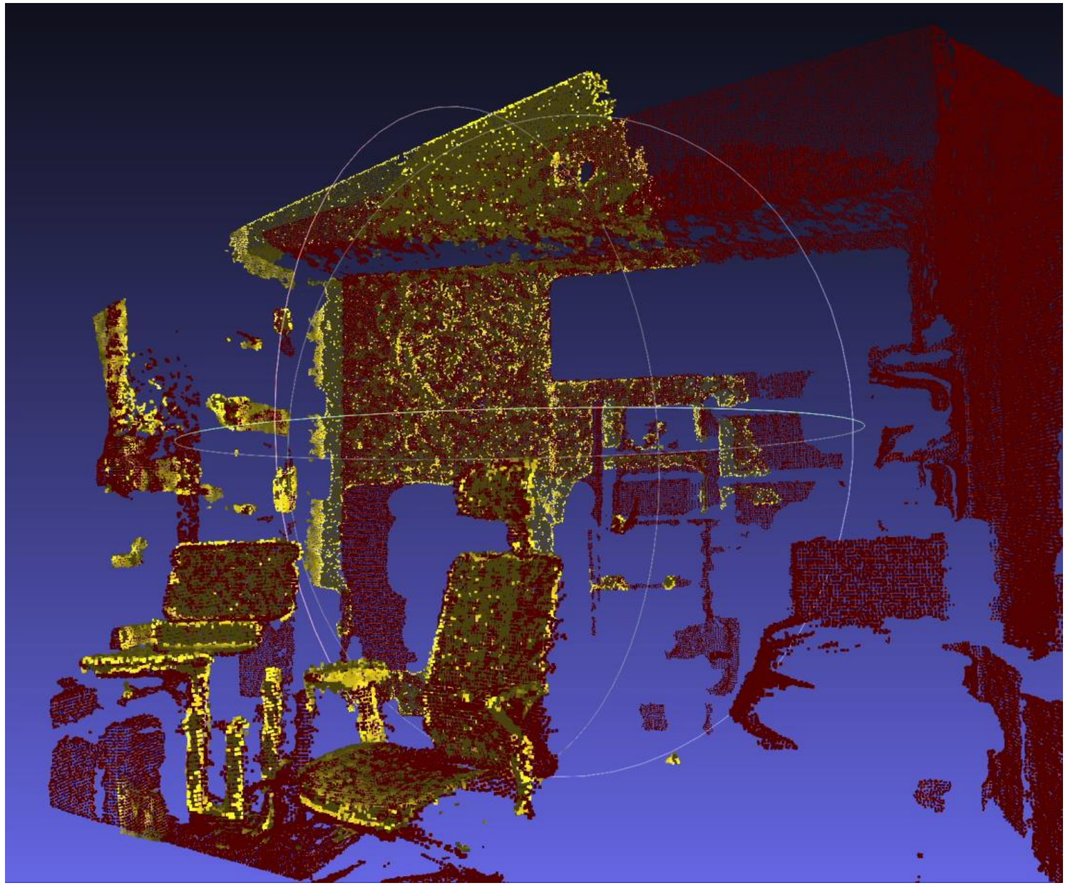


Colorized Point Cloud Reconstruction (Stereo Calib. + ICP)

ICP Ablation Study

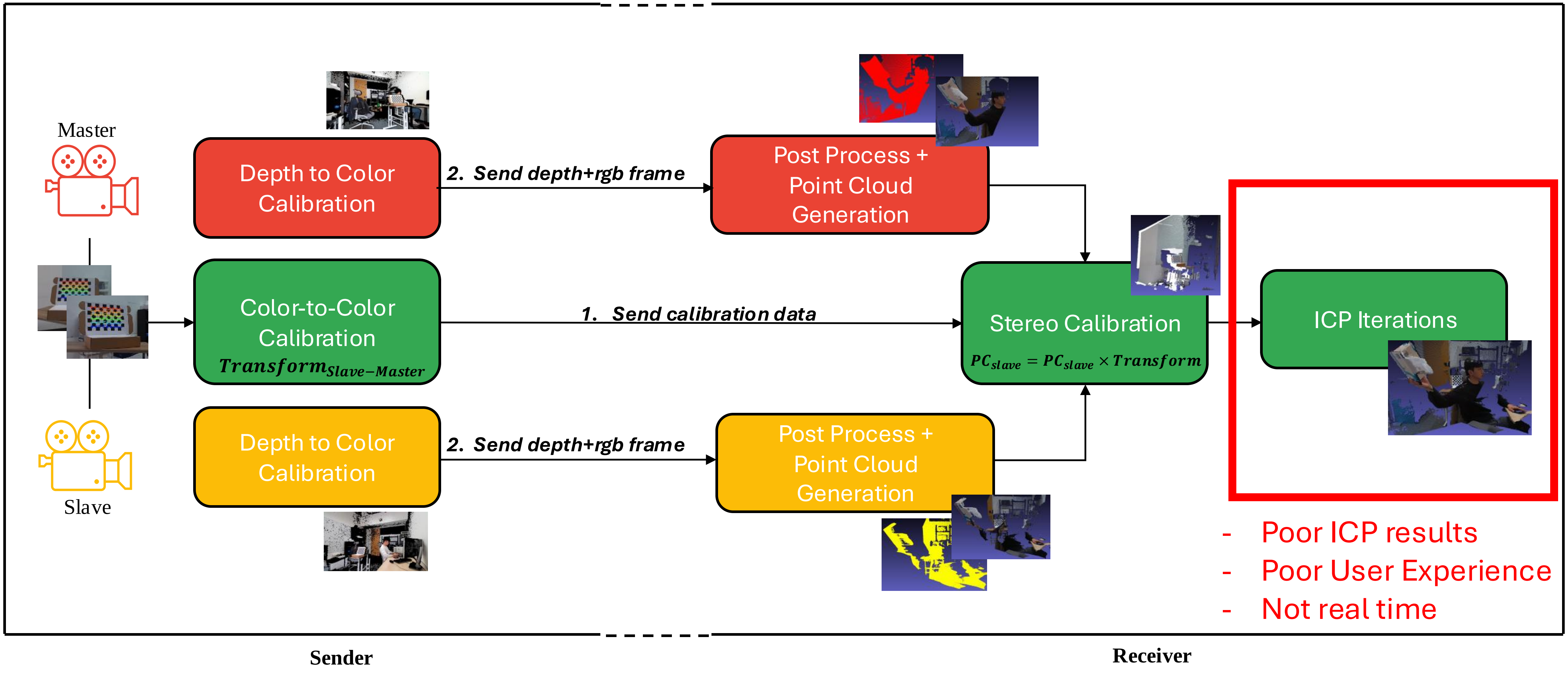


Point Cloud Reconstruction (No ICP)

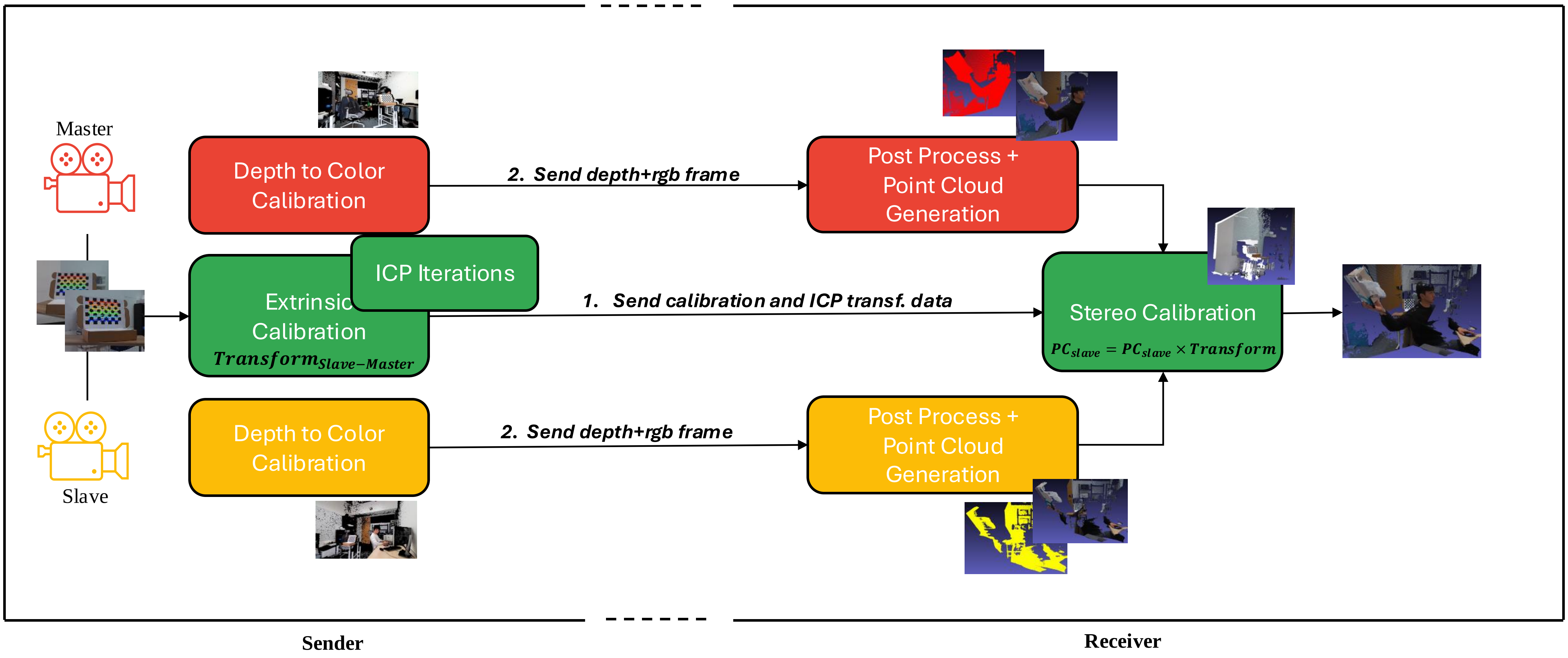


Point Cloud Reconstruction (With ICP)

System Design



System Design



Progress Update

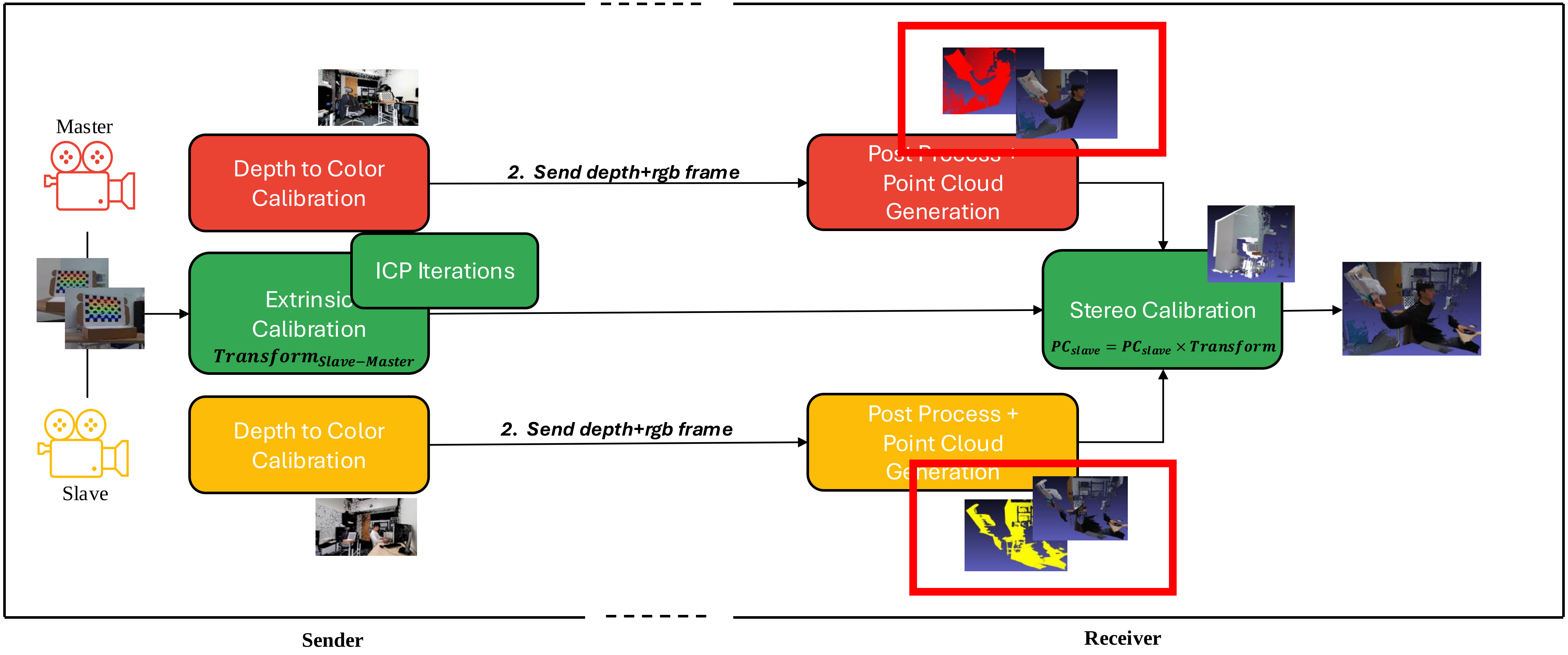
Progress from May to July 2024

Existing Problems in Current System

1. Problems caused by Fusion:
   - Problem#1: Some streams are randomly dropped (four streams needed but only two are live)
   - Problem#1.5: Fusion adds significant latency (especially for pointcloud generation)

2. Problems Persistent in BlenDR:
   - Problem#2: Flying pixels exist – poor appearance despite good RMSE

System Design



Problem#1: Dropping Streams

Two streams being correctly processed

some streams not being recognized

RTMP	#clients	Video				Audio				In bytes	Out bytes	In bits/s	Out bits/s	S
Accepted: 6		codec	bits/s	size	fps	codec	bits/s	freq	chan	7.98 MB	7.98 MB	5.29 Mb/s	5.3 Mb/s	
master														
live streams	2													
stream	2	H264	2.68 Mb/s	1920x1080	0		0 Kb/s			3.78 MB	3.78 MB	2.68 Mb/s	2.68 Mb/s	a

Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1224	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	4267	-4267	11s
1221	playing	143.248.57.176	LNX 9,0,124,2			0	4267	-4267	11s

RTMP	#clients
master_rgb	
live streams	0
slave_rgb	
live streams	0

RTMP	#clients	codec	bits/s	size	fps	codec	bits/s	freq	chan	In bytes	Out bytes	In bits/s	Out bits/s	S
slave														
live streams	2													
stream	2	H264	2.61 Mb/s	1920x1080	0		0 Kb/s			4.19 MB	4.19 MB	2.61 Mb/s	2.61 Mb/s	a

Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1225	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	4267	-4267	11s
1220	playing	143.248.57.176	LNX 9,0,124,2			0	4267	-4267	11s

Solution: Thread Scheduling

RTMP	#clients	Video				Audio				In bytes	Out bytes	In bits/s	Out bits/s	State	Time
Accepted: 14		codec	bits/s	size	fps	codec	bits/s	freq	chan	32.31 MB	32.31 MB	0 Kb/s	0 Kb/s		14h 52m 26s
master															
live streams	2														
stream	2	H264	0 Kb/s	1920x1080	0		0 Kb/s			2.3 MB	2.3 MB	0 Kb/s	0 Kb/s	active	8s

Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1235	playing	143.248.57.176	LNX 9,0,124,2			0	2400	-2400	8s
1231	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	2400	-2400	8s

master_rgb															
live streams	2														
rgb_stream	2	H264	0 Kb/s	640x360	0		0 Kb/s			269 KB	269 KB	0 Kb/s	0 Kb/s	active	8s

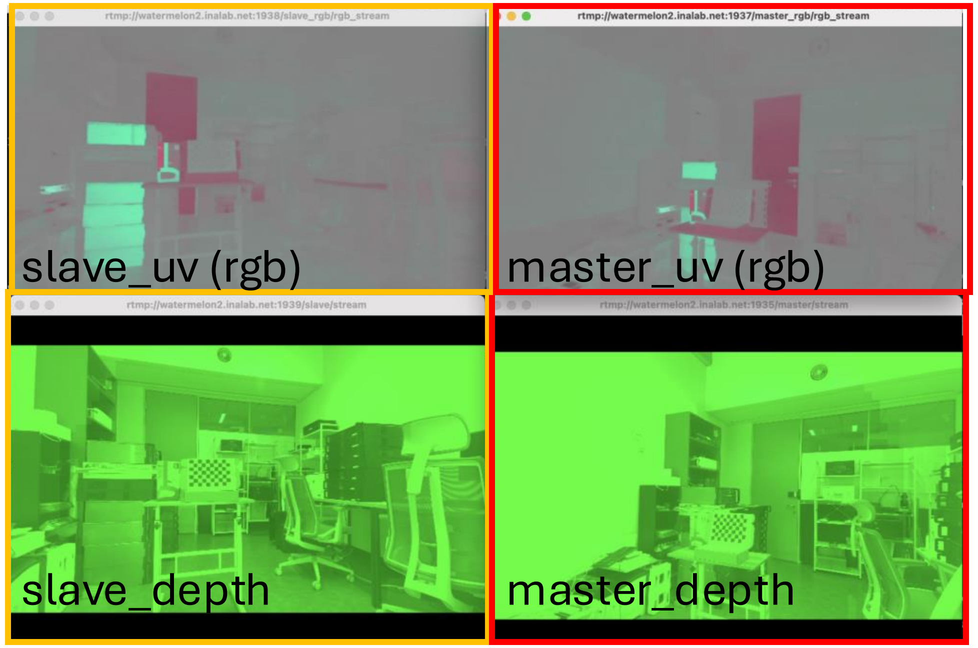
Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1236	playing	143.248.57.176	LNX 9,0,124,2			0	2367	-2367	8s
1232	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	2367	-2367	8s

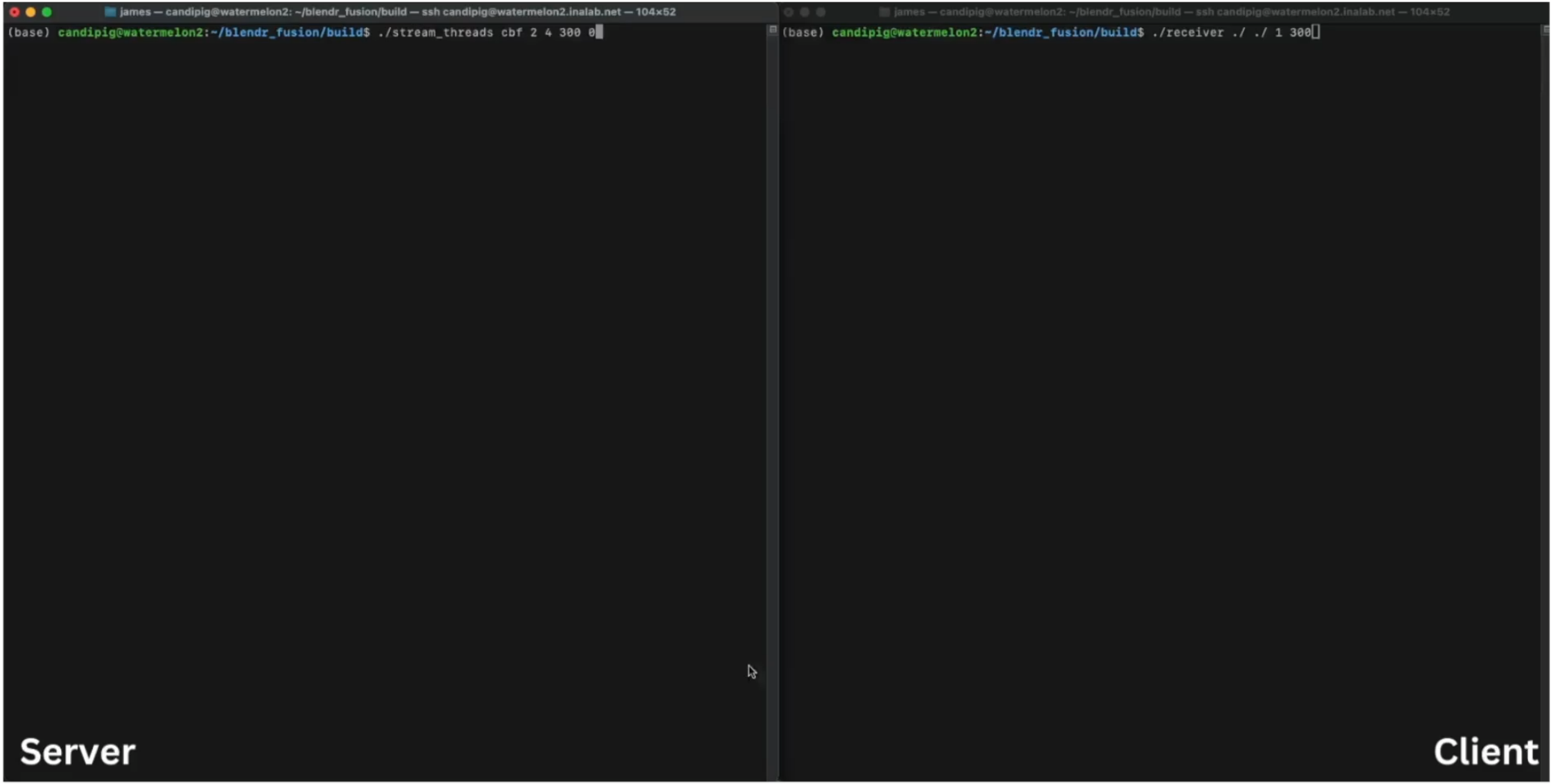
slave_rgb															
live streams	2														
rgb_stream	2	H264	0 Kb/s	640x360	0		0 Kb/s			276 KB	276 KB	0 Kb/s	0 Kb/s	active	8s

Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1238	playing	143.248.57.176	LNX 9,0,124,2			0	2433	-2433	8s
1234	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	2433	-2433	8s

slave															
live streams	2														
stream	2	H264	0 Kb/s	1920x1080	0		0 Kb/s			2.26 MB	2.26 MB	0 Kb/s	0 Kb/s	active	8s

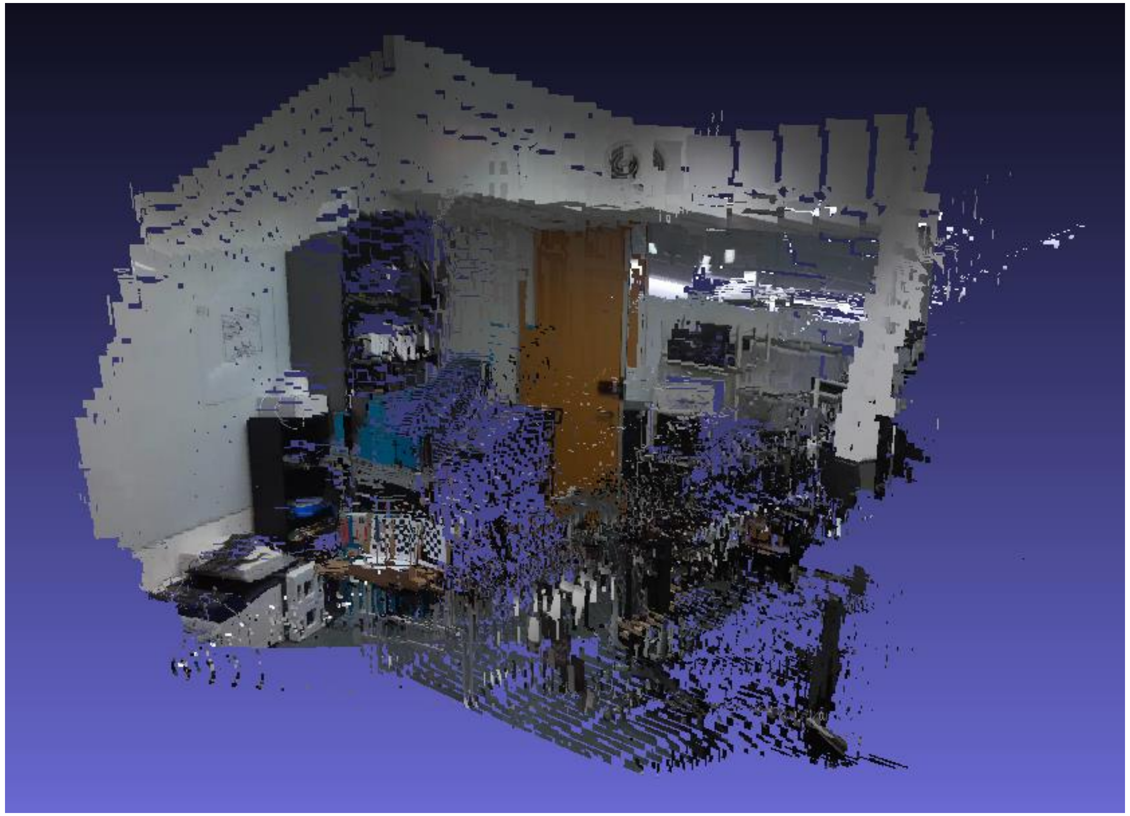
Id	State	Address	Flash version	Page URL	SWF URL	Dropped	Timestamp	A-V	Time
1237	playing	143.248.57.176	LNX 9,0,124,2			0	2400	-2400	8s
1233	publishing	143.248.57.176	FMLE/3.0 (compatible; Lavf57.83			0	2400	-2400	8s

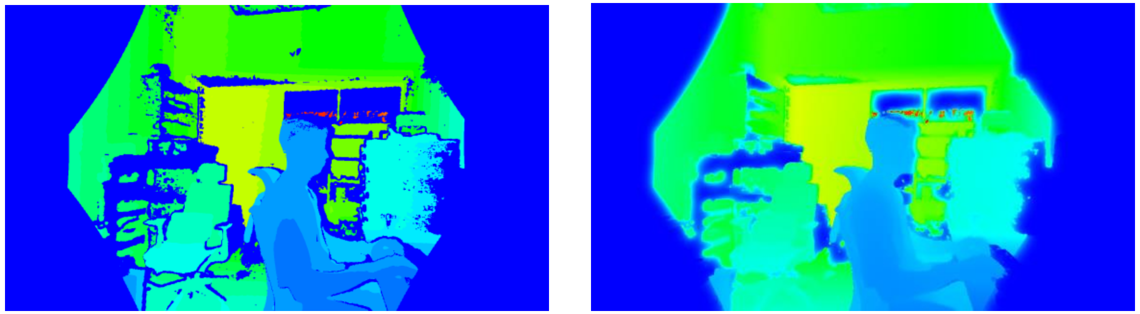



james — candipig@watermelon2: ~/blendr_fusion/build — ssh candipig@watermelon2.inalab.net — 104×52
(base) candipig@watermelon2:~/blendr_fusion/build$ ./stream_threads cbf 2 4 300 0
james — candipig@watermelon2: ~/blendr_fusion/build — ssh candipig@watermelon2.inalab.net — 104×52
(base) candipig@watermelon2:~/blendr_fusion/build$ ./receiver ./ ./ 1 300
Server
Client

Problem#2: Flying Pixels

- Flying Pixel Effect Remains
  - False depth values being added for continuity during **depth-filling**





Depth Filling

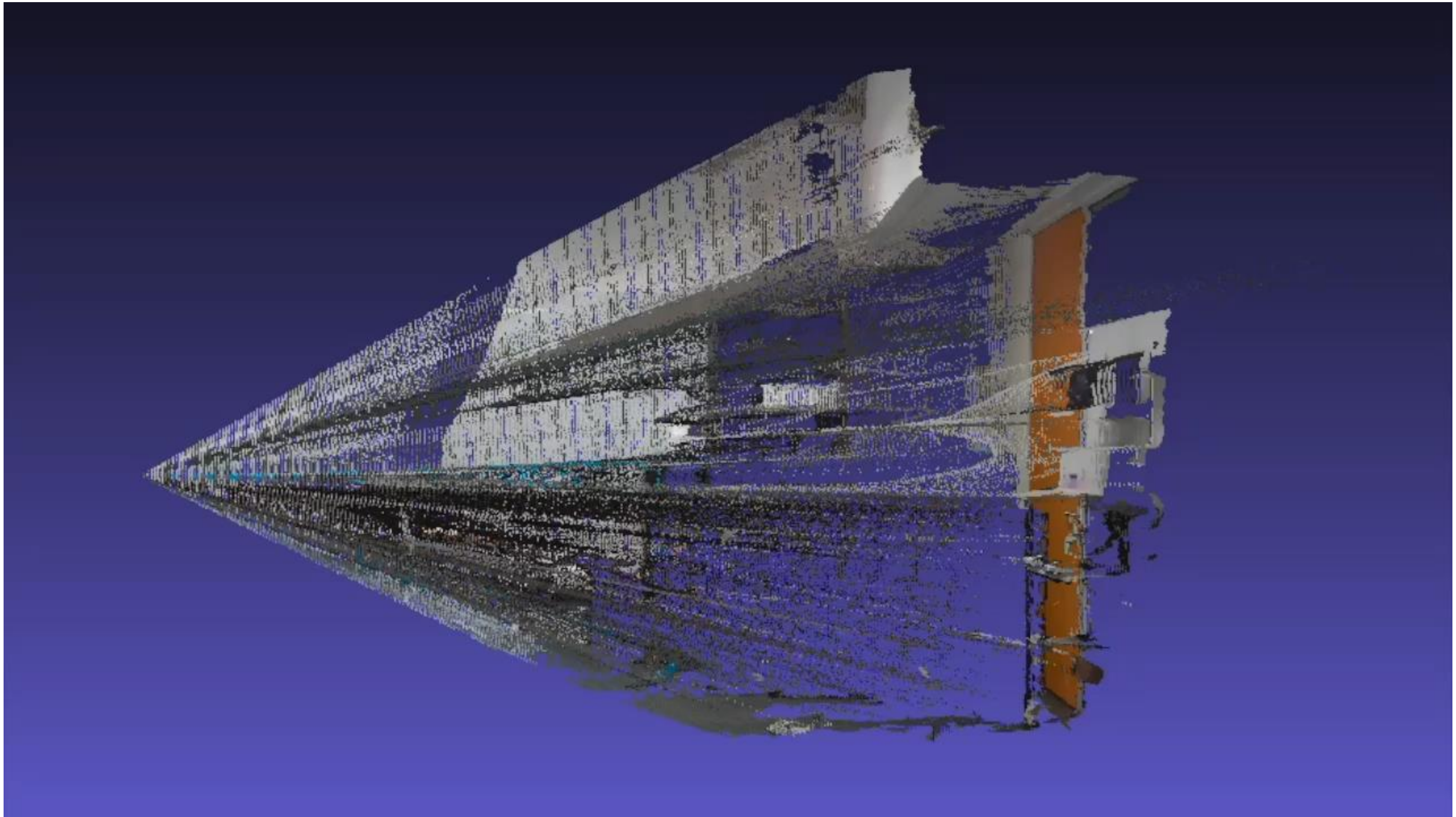
Solution: Corrected Post-Processing

- **Identified Problem:** Threshold Mask is simply subtracted from the received depth image

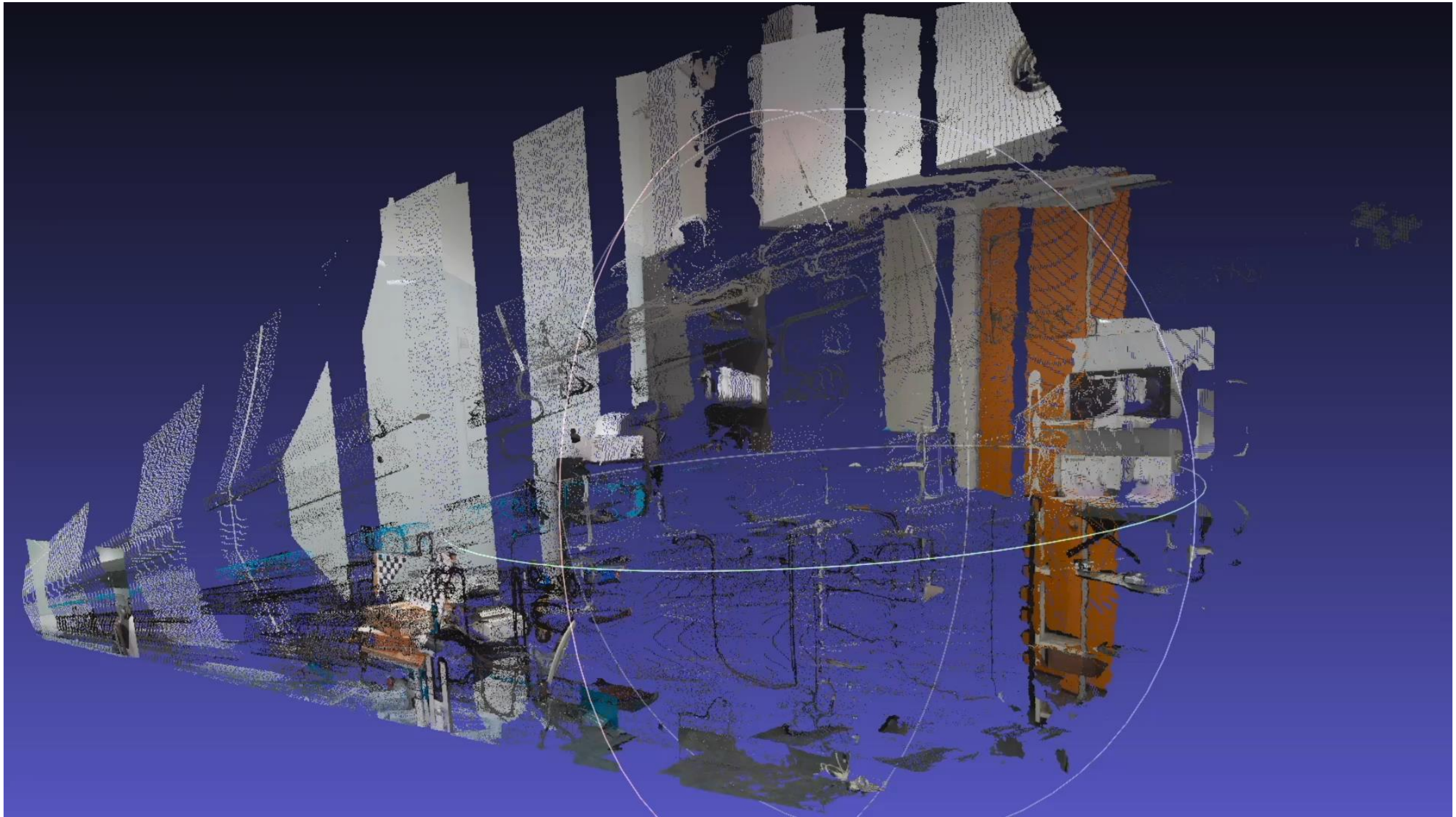
```
# thr_result is matrix with 1 and 0 indicating edge
inverted_thr_result = 1 - thr_result
depth_with_edges_removed = decoded_depth * inverted_thr_result
```

- Additional Improvements:
  - Post-processing using O3D (remove_statistical_outlier) [5][6][7][8]
    - Other options: remove_radius_outlier [8]

Ablation Study Results: Ours



Ablation Study Results: Triangle Method



New Contributions to BlenDR

- End to End System that allows for multi-view fusion
- Effectively Remove Flying Pixels
- Comparison with GROOT (PointCloud Compression Method)
  - Better ground-truth similarity compared to GROOT
  - Results:
    - Without Fusion: o3d HD (cm): **3.49 (groot)**, 19.40 (triangle), **2.28 (ours)**
    - With Fusion: o3d HD (cm): **4.53 (groot),** 16.35 (triangle), **2.76 (ours)**
- Potential addition: User experience studies

Additional Progress Done

- Latency minimization – Problem#1.5
  - Thread scheduling and GPU Optimization (200ms cut down to 60ms)

- Modularization of code
  - Fusion class made for easy usage
  - Automatize recording, depth filling, depth packing, and point cloud generation

```cpp
namespace Fusion
{
    static std::mutex main_pc_mutex;
    static std::mutex slave_pc_mutex;

    class Fusion
    {
    public:
        float chessboard_square_length = 0.;                   // must be included in the input params
        int32_t color_exposure_usec = 8000;                    // somewhat reasonable default exposure time
        int32_t powerline_freq = 2;                            // default to a 60 Hz powerline
        cv::Size chessboard_pattern;                           // height, width. Both need to be set.
        uint16_t depth_threshold = 15000;                      // default to 1 meter
        double calibration_timeout = 60.0;                     // default to timing out after 60s of trying to get calibrated
        double duration = std::numeric_limits<double>::max(); // run forever
        size_t num_devices = 2;
        vector<uint32_t> device_indices{0};

        // for calibration
```

```cpp
    namespace SingleCPU
    {
        void PreparePointCloud(Fusion& fusion);
        void CreatePointCloud(Fusion& fusion);
    }

    namespace MultiThread
    {
        void CreatePointCloud(Fusion& fusion, int max_threads);
    }

    namespace GPU
    {
        void AllocMemory(Fusion& fusion);
        void PreparePointCloud(Fusion& fusion);
        void CreatePointCloud(Fusion& fusion);
    }
```

Future Goal and Plan

Date	Task
JUL. 25 – AUG. 12	• Conduct Experiments for Fusion Evaluation • Optimize code for real-time point cloud fusion • Automate test benches for modularization and ease of use
AUG. 13	• Plane to Texas
AUG – SEP. 12	• Paper Writing and Additional Data Retreival
SEP. 12	• NSDI ‘25 Paper Abstract Due
SEP. 19	• NSDI ‘25 Full Paper Due

Thank you.

Reference

[1] https://scholarworks.calstate.edu/downloads/qr46r322x?locale=it

[2] https://learn.microsoft.com/en-us/azure/kinect-dk/coordinate-systems

[3] https://ieeexplore.ieee.org/document/7335499

[4] https://www.open3d.org/docs/release/tutorial/pipelines/colored_pointcloud_registration.html

[5] https://www.mdpi.com/1424-8220/21/2/664

[6] https://www.e-consystems.com/blog/camera/technology/what-is-flying-pixel-and-how-can-it-be-mitigated-in-3d-imaging-for-time-of-flight-cameras/

[7] https://www.mdpi.com/1424-8220/21/14/4628

[8] https://www.open3d.org/docs/0.12.0/tutorial/geometry/pointcloud_outlier_removal.html

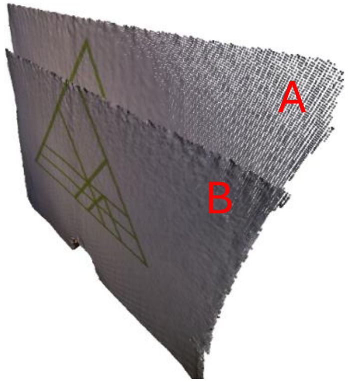
Iterative Closest Point Algorithm (ICP)

- Summary of ICP (Colored)[6]
  1. Start with initial guess transformation, T^0
  2. For each point in point cloud, find correspondence points, *K*, based on both spatial proximity and color similarity.

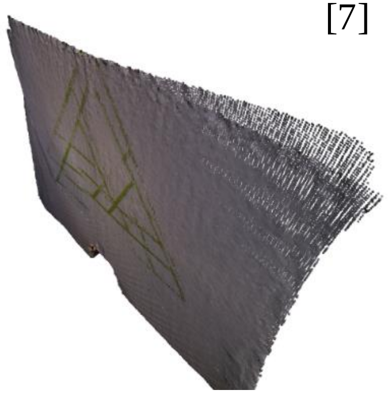
     → Use Euclidean distance for difference
  3. Calculate the transformation that minimizes a cost function (Least-Squares Fitting Function)
  4. Apply this transformation to the source point cloud and repeat until convergence or until maximum number of iterations.

Function used:
open3d::pipelines::registration::RegistrationColoredICP()

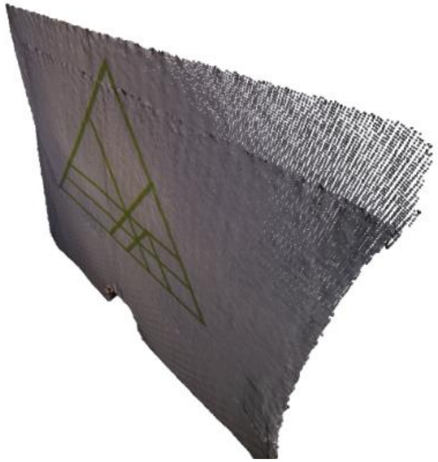
[7]



Point-to-Point ICP



Point-to-Plane ICP



Colored ICP