“Kinect Fusion: HW Implementation and Acceleration of a Dense SLAM Algorithm” Final Report

ITRI EOSL Summer Intern Program

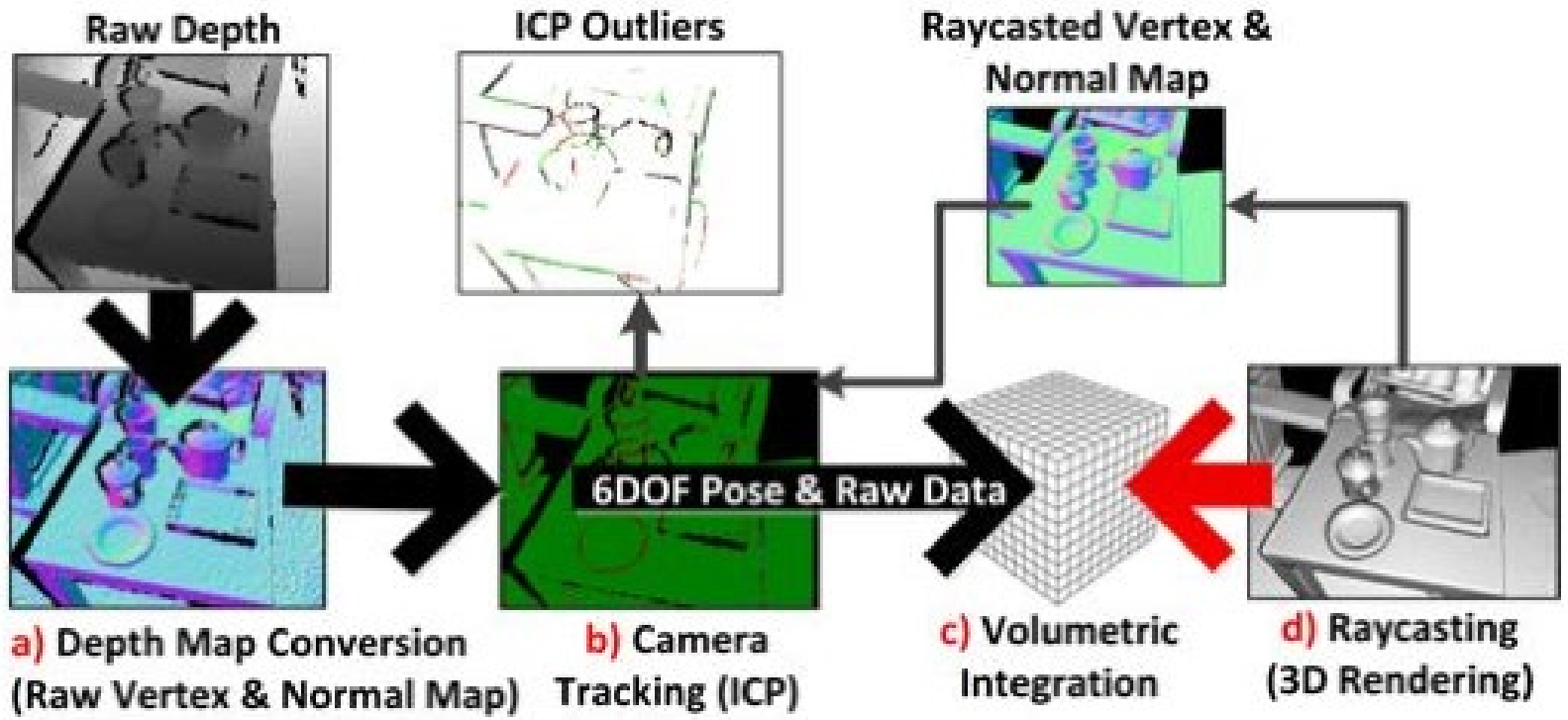
李承澔

NTHU EECS24 109020014

Kinect Fusion

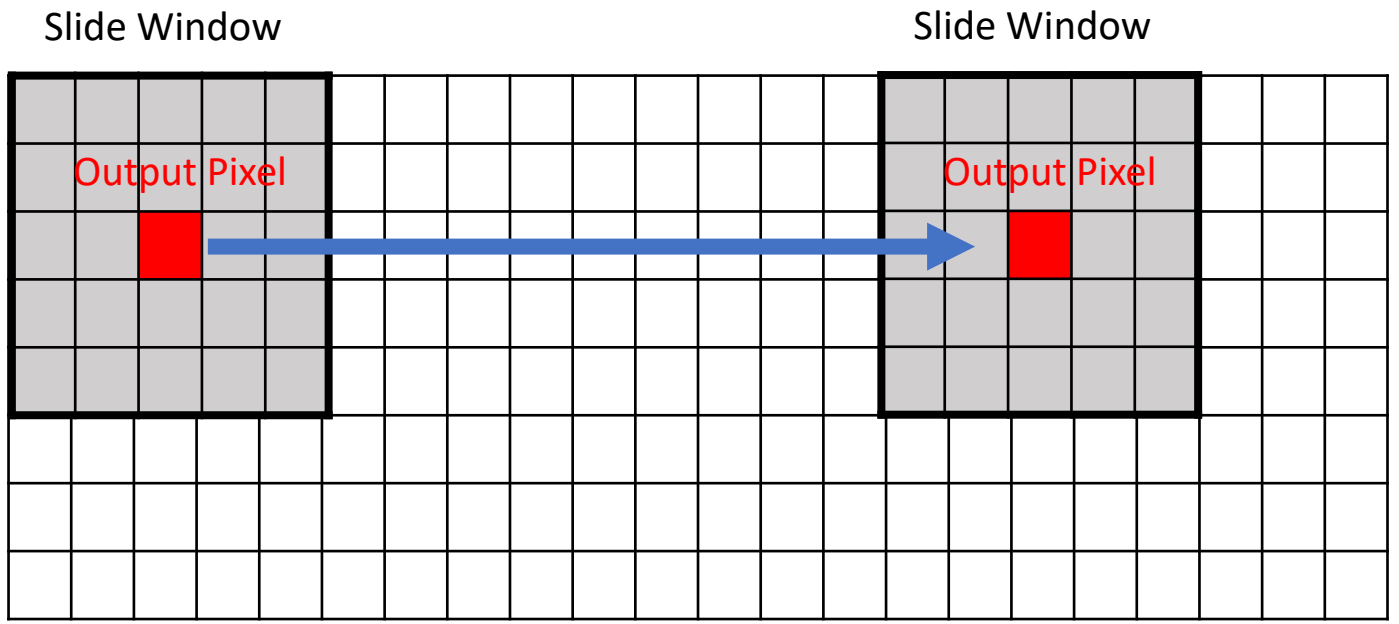
Kinect Fusion is an algorithm that reconstructs a 3D environment from a depth map acquired either by a depth camera or a stereo camera setup

It follows the workflow:



a) Depth Map Conversion – Bilateral Filter Kernel

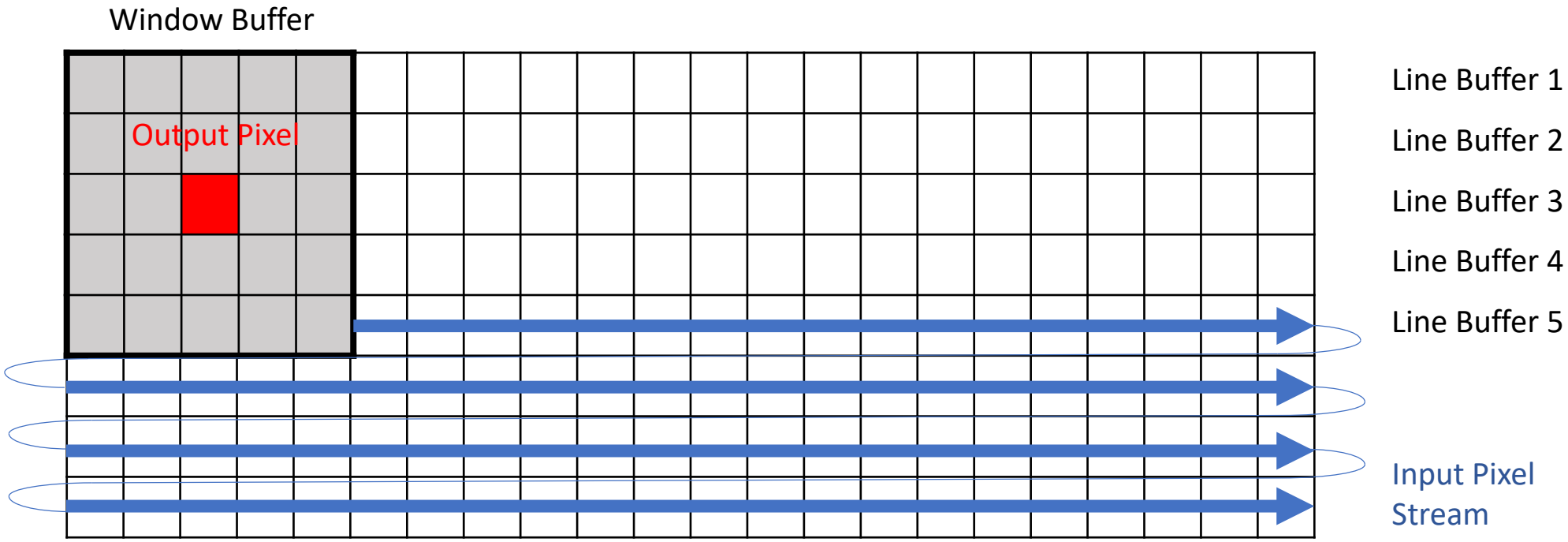
The algorithm employs a bilateral filter to reduce noise in the input depth map. This filter uses a 5x5 sliding window.



a) Depth Map Conversion – Bilateral Filter Kernel Optimize

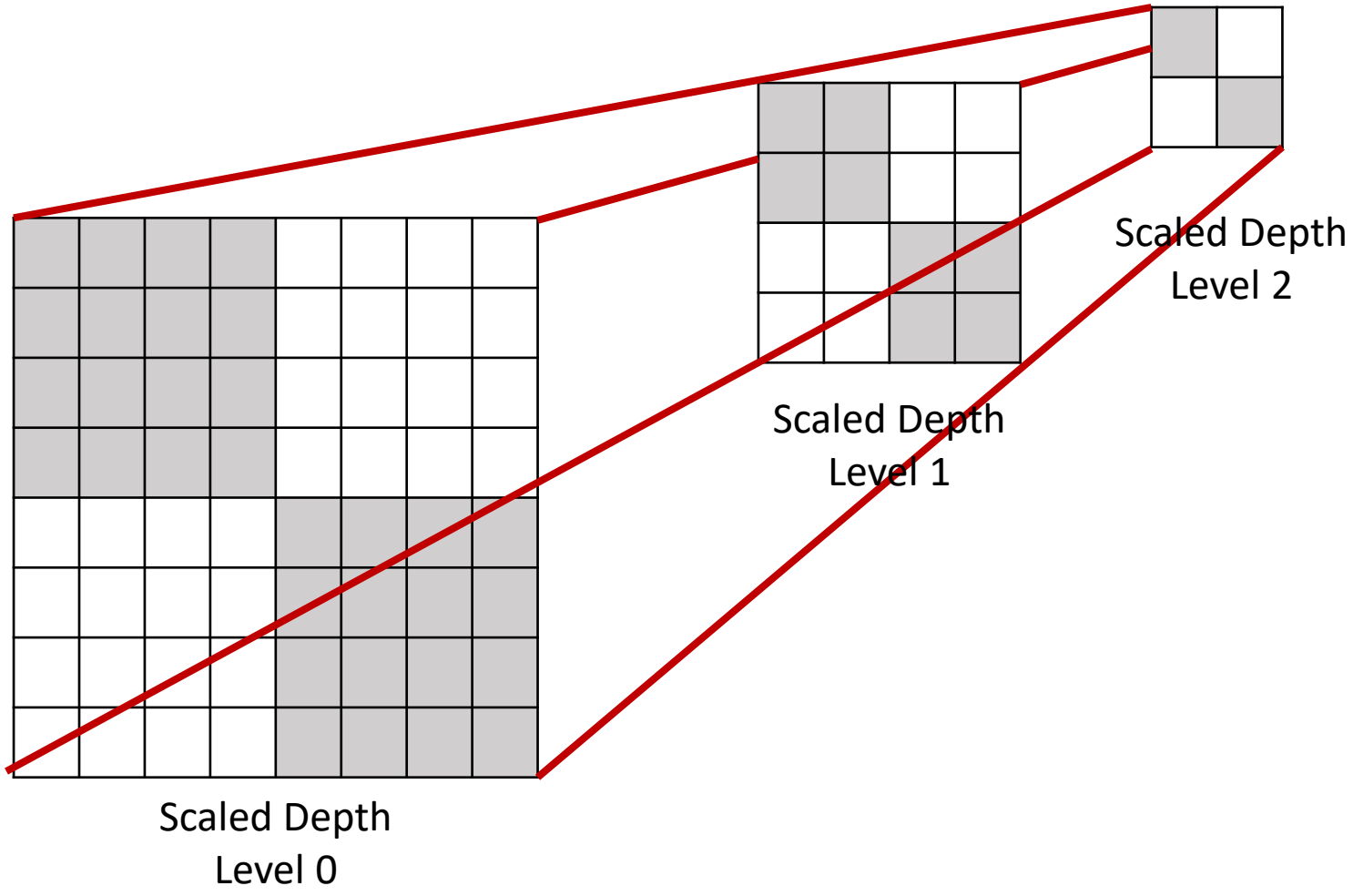
The use of a 5x5 sliding window results in numerous read operations—25 memory read operations for each pixel computation, exactly.

To mitigate this, we have implemented line and window buffers in hardware to cache pixels and reduce the number of read operations per pixel computation.



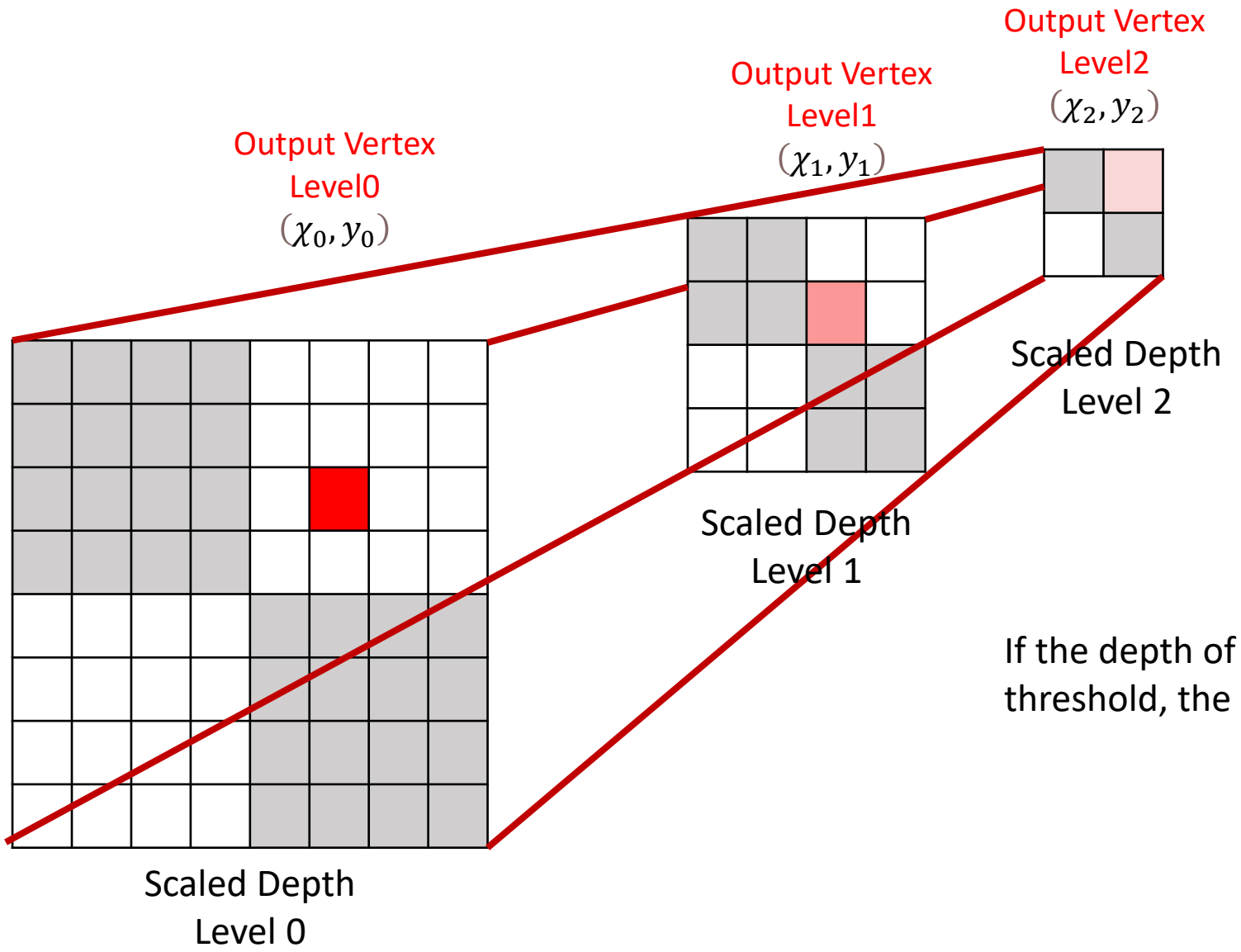
a) Depth Map Conversion – Half Sample Kernel

After filtering, Kinect Fusion constructs a scale pyramid and prepares scaled-down depth maps for subsequent vertex and normal computations.



a) Depth Map Conversion – Depth 2 Vertex

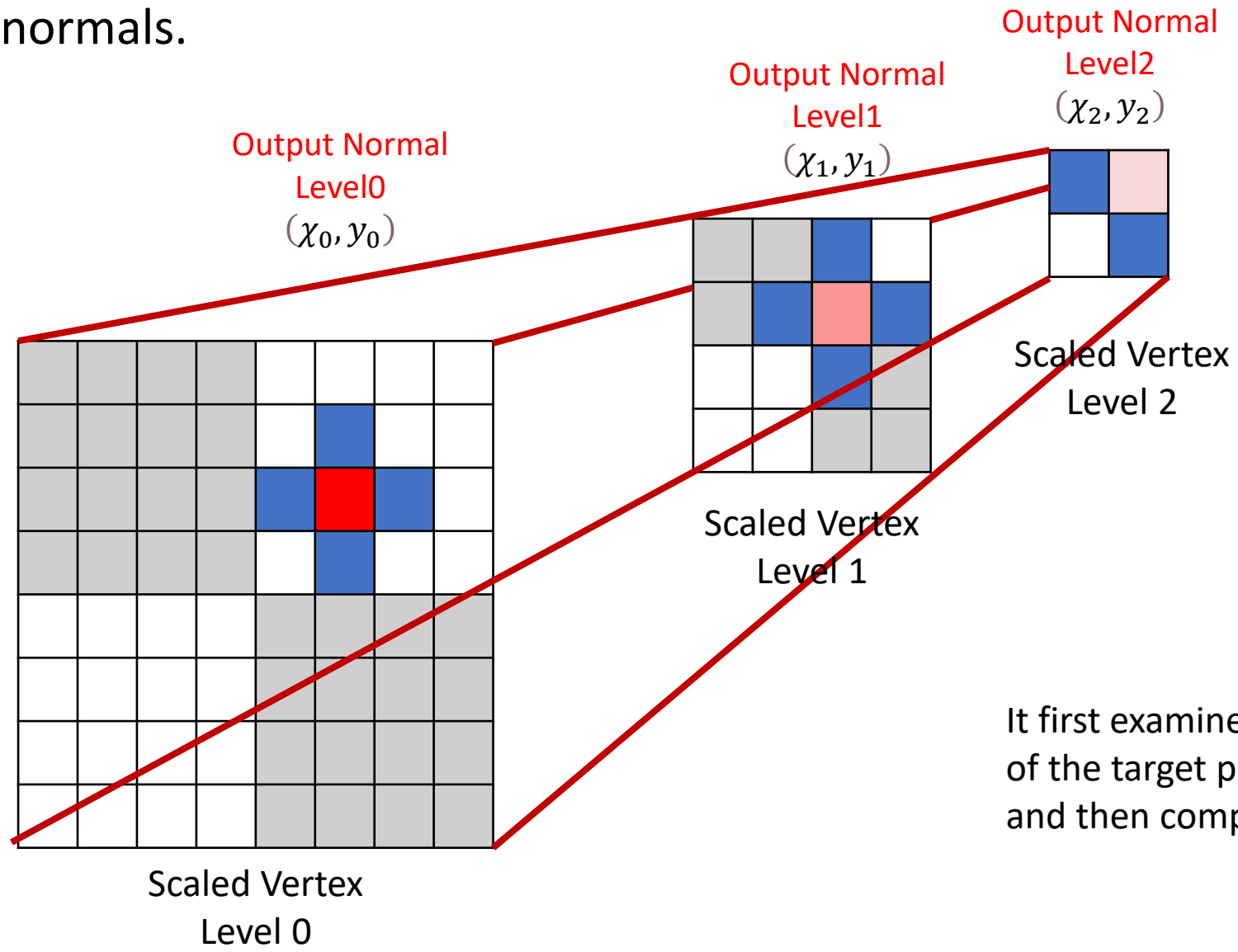
Kinect Fusion traverses all three scaled depth maps to produce three sets of vertices.



If the depth of the target pixel exceeds a predefined threshold, the pixel is annotated as a vertex

a) Depth Map Conversion – Vertex 2 Normal

The algorithm also traverses all three scaled vertex maps to produce three sets of normals.

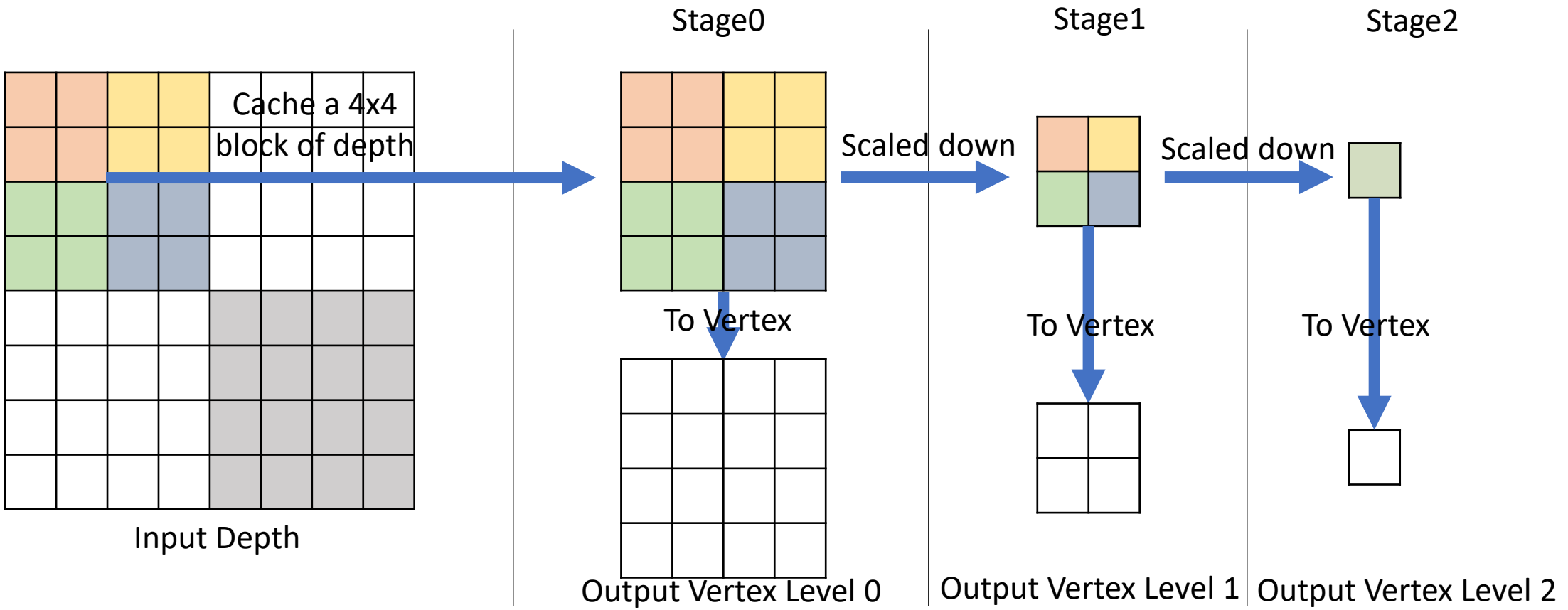


It first examines the neighbors (up, down, left, right) of the target pixel,
and then computes the pixel's normal

a) Depth Map Conversion – Half Sample Depth 2 Vertex Optimize

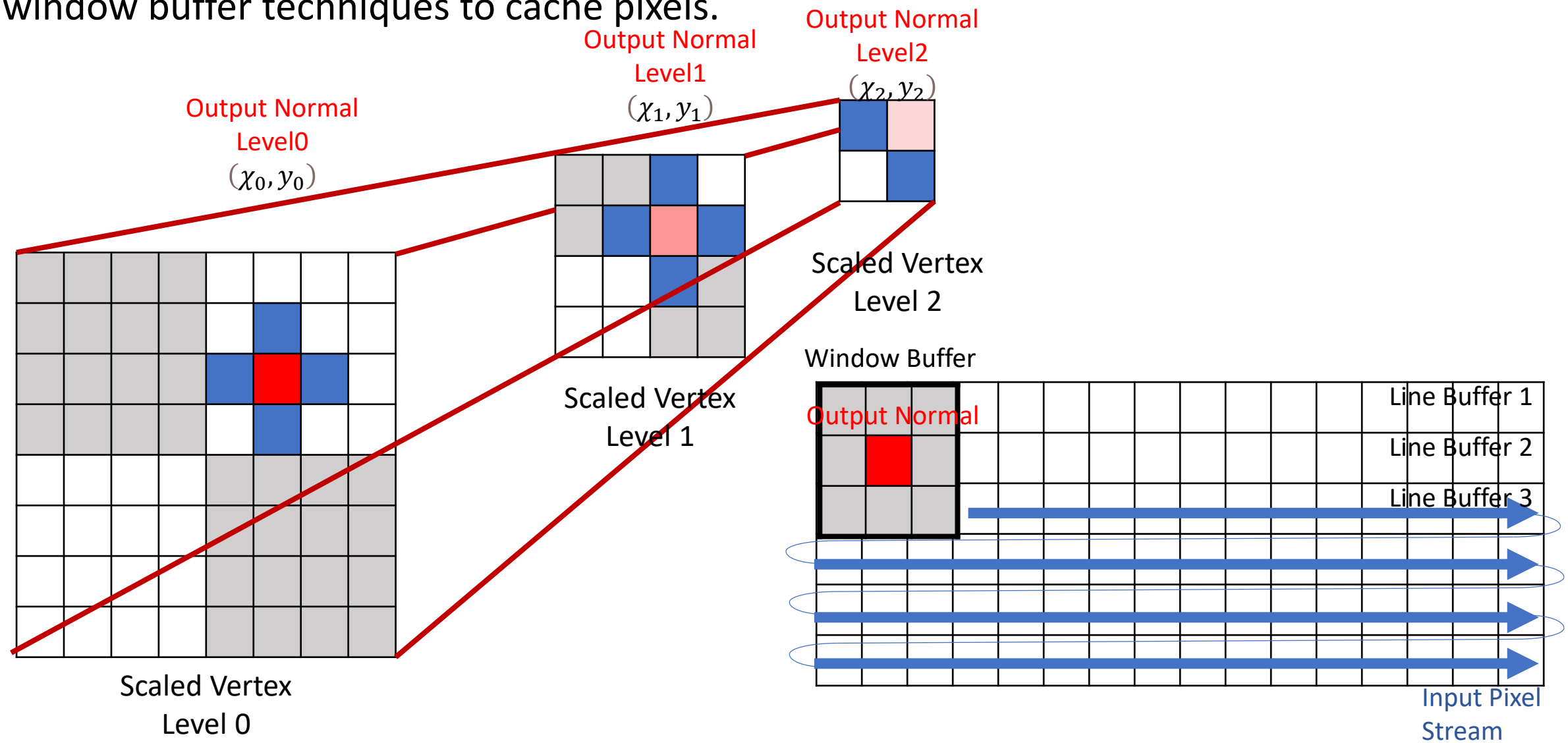
Since Kinect Fusion first generates scaled depth maps and then traverses each one to produce vertices, these two operations can be merged.

We read a 4x4 block of pixels and leverage parallel computing resources to calculate the next-level depth and its corresponding vertex.



a) Depth Map Conversion – Vertex 2 Normal Optimize

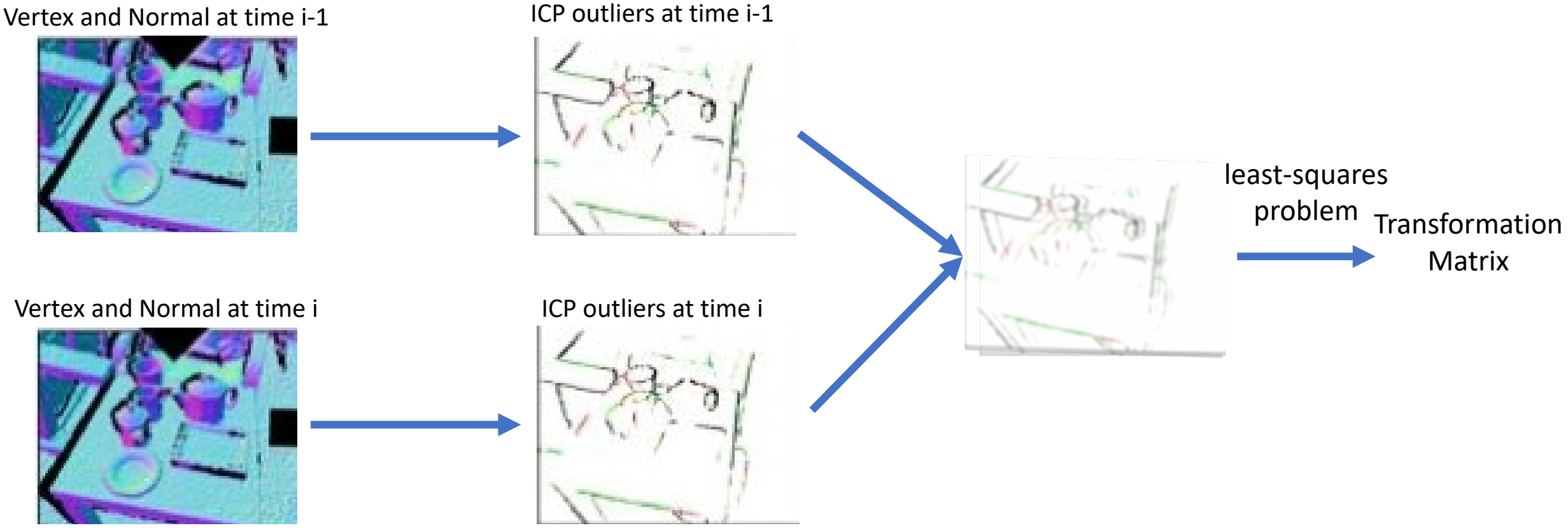
Since the vertex-to-normal conversion also utilizes a sliding window to consider the neighboring pixels of the target and compute normals, we continue to apply line and window buffer techniques to cache pixels.



b) Camera Tracking – ICP Algorithm

Kinect Fusion traverses all pixels and matches vertices based on their depth and normal.

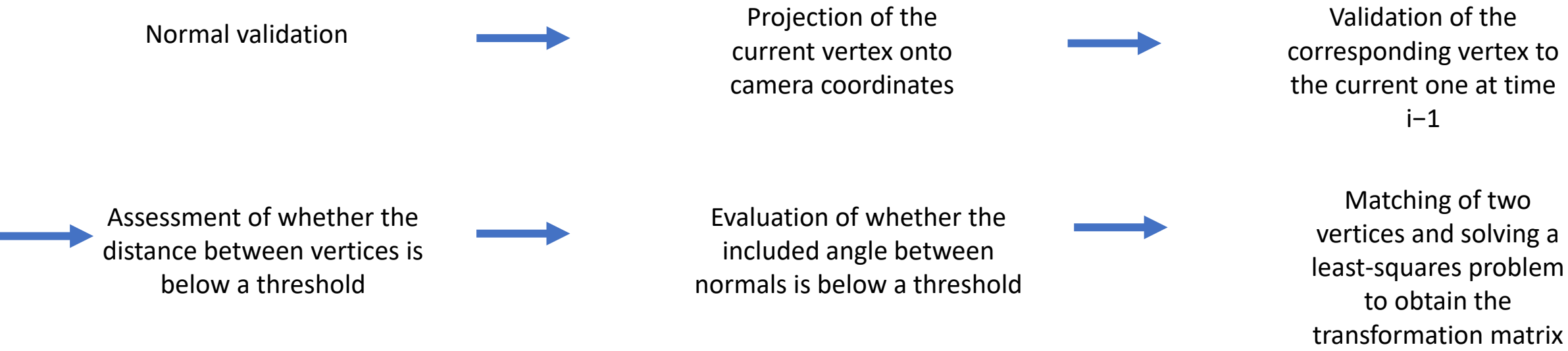
The algorithm employs an iterative approach to solve a least-squares problem, ultimately yielding the camera's transformation matrix.



b) Camera Tracking – ICP Algorithm Optimize

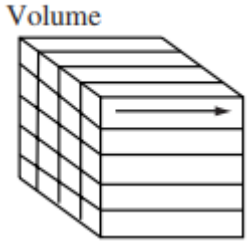
Given that the tracking algorithm demands substantial computational resources,

we have divided it into multiple stages and pipelined them to achieve an initiation interval (II) of 1.



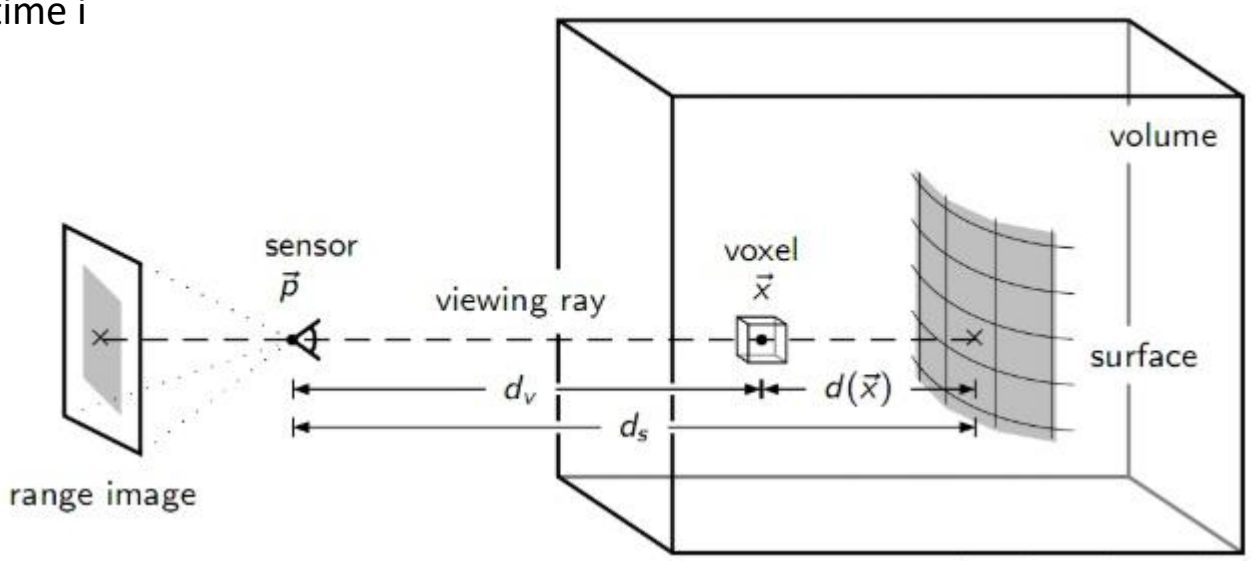
c) Volumetric Integration

Kinect Fusion updates the 3D voxels based on the current camera position and vertex depth. The algorithm performs the following steps:



Traverses a row of voxels along the z-axis at time i

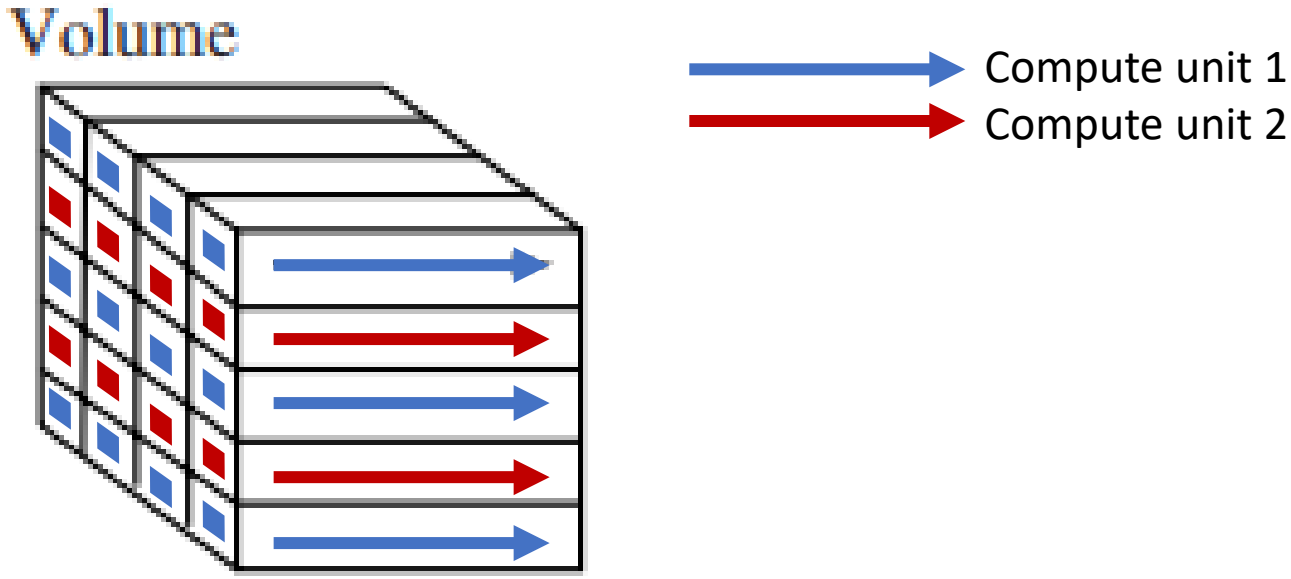
Transforms the voxel through the coordinate transformation: global → camera → image



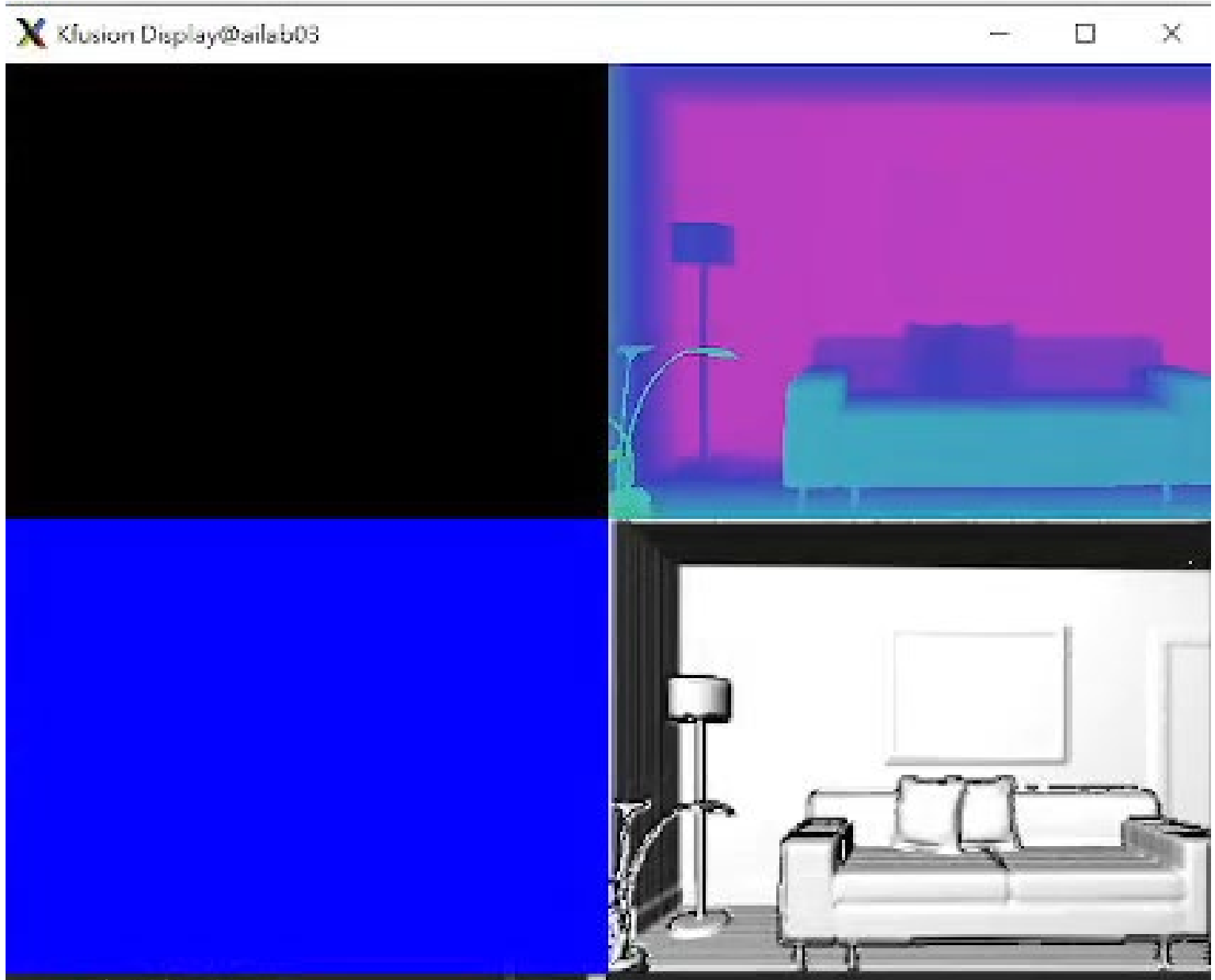
Updates the entire row of voxels based on the pixel depth, which is transformed from the voxel

c) Volumetric Integration - Optimization

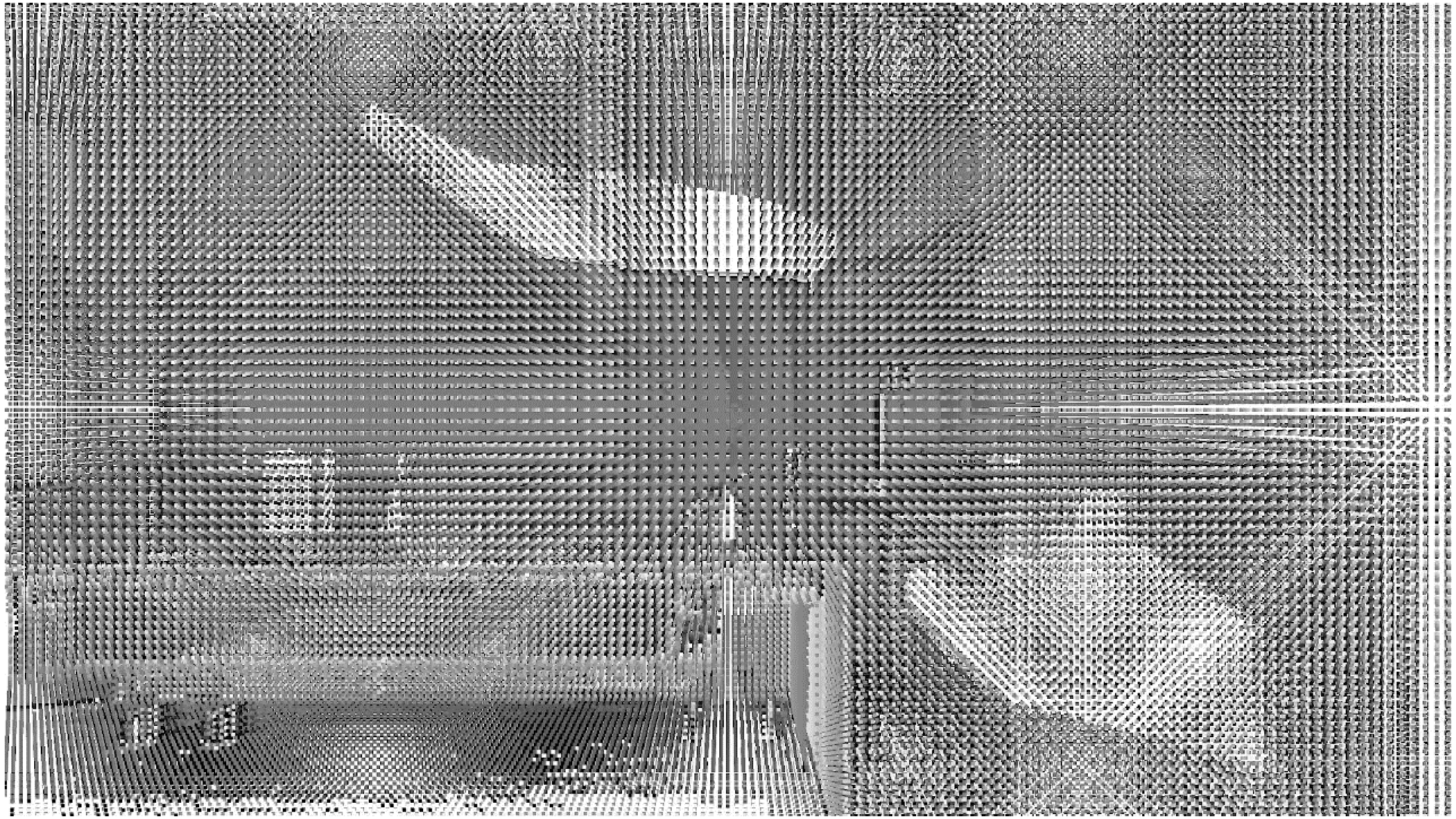
Given that the integration traverses all voxels in the 3D space, resulting in heavy computation, we deploy two compute units to parallelly process two rows of voxels along the z-axis.



Result and Compare – Reconstruction Process



Result and Compare – Volumetric Reconstruction



Result and Compare

SW : Intel(R) Xeon(R) E-2288G CPU @ 3.70GHz OpenMP disable

HW : AMD Alveo™ U50 FPGA

Kernel Average Latency(ms)	Bilateral Filter Kernel	Half Sample Kernel	Depth2 Vertex Kernel	Vertex2 Normal Kernel	Track Kernel	Integrate Kernel	Error (ATE)
SW	26.504	36.602	53.494	51.648	584.756	10.841	18.069
HW Baseline	43.374	31.789		0.738	498.939	3149	17.280
HW Optimize	0.516	31.789		0.541	1.236	29.470	17.280

Result and Compare

Resource Usage &Timing		Bilateral Filter Kernel	HalfSample 2Vertex Kernel	Vertex2 Normal Kernel	Track Kernel	Integrate Kernel
Frequency (MHz)	baseline	379.794	379.794	411.015	348.918	351.370
	optimize	411.015	379.794	411.015	327.332	411.015
II	baseline	=1	=1	violation	=1	=1
	optimize	=1	=1	=1	=1	=1
BRAM (%)	baseline	0	0	0	0	0
	optimize	0	0	0	4	16
DSP (%)	baseline	~0	~0	~0	~0	~0
	optimize	1	~0	~0	1	4
FF (%)	baseline	~0	~0	~0	4	1
	optimize	1	~0	~0	2	3
LUT (%)	baseline	1	3	1	24	3
	optimize	5	3	2	3	6