1. Parallel seems to always make thing slower

#pragma omp parallel

{

#pragma omp single

{

while (centerCubeI < 3)

{

std::cout << "loop 1" << std::endl;

#pragma omp parallel for num\_threads(5)

for (int k = 0; k < \_laserCloudDepth; k++)

{

for (int j = 0; j < \_laserCloudHeight; j++)

{

// std::cout << "before rotate: " << \_laserCloudCornerArray[\_laserCloudWidth - 1] << std::endl;

for (int i = \_laserCloudWidth - 1; i >= 1; i--)

{

const size\_t indexA = toIndex(i, j, k);

const size\_t indexB = toIndex(i - 1, j, k);

std::swap(\_laserCloudCornerArray[indexA], \_laserCloudCornerArray[indexB]);

std::swap(\_laserCloudSurfArray[indexA], \_laserCloudSurfArray[indexB]);

}

const size\_t indexC = toIndex(0, j, k);

\_laserCloudCornerArray[indexC]->clear();

\_laserCloudSurfArray[indexC]->clear();

}

}

centerCubeI++;

\_laserCloudCenWidth++;

}

} // omp single

} // omp parallel

Code slows down by 2x

1. Std rotate doesn’t help because the 3D array is stored as a flattened 1D array. Thus, the rotations are highly nontrivial in two dimensions.