Milestone 3:

3D Descriptor for Object Detection

Yu Sun • 12.3.2018

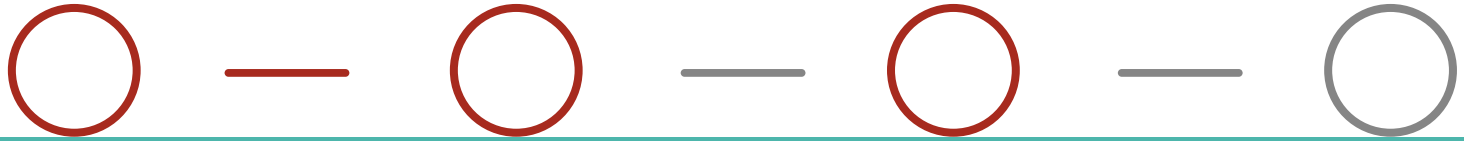
3D Descriptor for Object Detection

Recent progress

- 3D-Descriptor (SHOT) implementation
- Radius Search
- Correspondance Search

- Optimization

Project Recap



Downsample the point cloud (features)

Compute Normals

Compute Descriptor

Local reference frame

Use normals and color information

Descriptor Correspondence

Rejection by score

Grouping

Compute transformation

Progress SHOT

Color-SHOT

- Chain signatures of histogram relative to different <property, measurement> pairs
- Color and Normals
- RGB to CIELAB color space (abs difference)
- Dot product for normals (local reference frame · every neighbor within radius)
- Interpolate both channel between neighbors => descriptor of length 352 (Not implemented yet)

F. Tombari, S. Salti, L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching", *IEEE International Conference on Image Processing (ICIP)*, September 11-14, Brussels, Belgium, 2011.

Progress - SHOT (Local Reference)

Refined Normal Estimation

- Weighted Total Least Square **M** (Distant from Feature)
- Eigenvalue Decomposition of covariance matrix **M**
- Covariance Matrix calculated using neighbors within pre-set radius

Disambiguation of directions

- Eigenvectors with decreasing eigenvalues
- Choose sign of eigenvectors coherent with majority of vectors it’s representing

F. Tombari *, S. Salti *, L. Di Stefano, "Unique Signatures of Histograms for Local Surface Description", *11th European Conference on Computer Vision (ECCV)*, September 5-11, Hersonissos, Greece, 2010.

Radius Search

- For each feature, -/+ the radius to get the max/min grid indice to search
- Store the value into 6 * N_features into shared memory (u_int8 * ~ 6 * 3000)
- Use AtmoicAdd with global memory to count number of neighbors for each indices
- Record distance to feature

KDTree Searching

- To find the correspondence between scene and model descriptor
- Descriptor has 352 dimension so KDTree is best for searching
- Tree construction is in CPU
- Tree is represented as an array in GPU
- Search uses index rather than pointer
- Not yet tested because shot has interpolation not done

Optimization (Uniform Sampling)

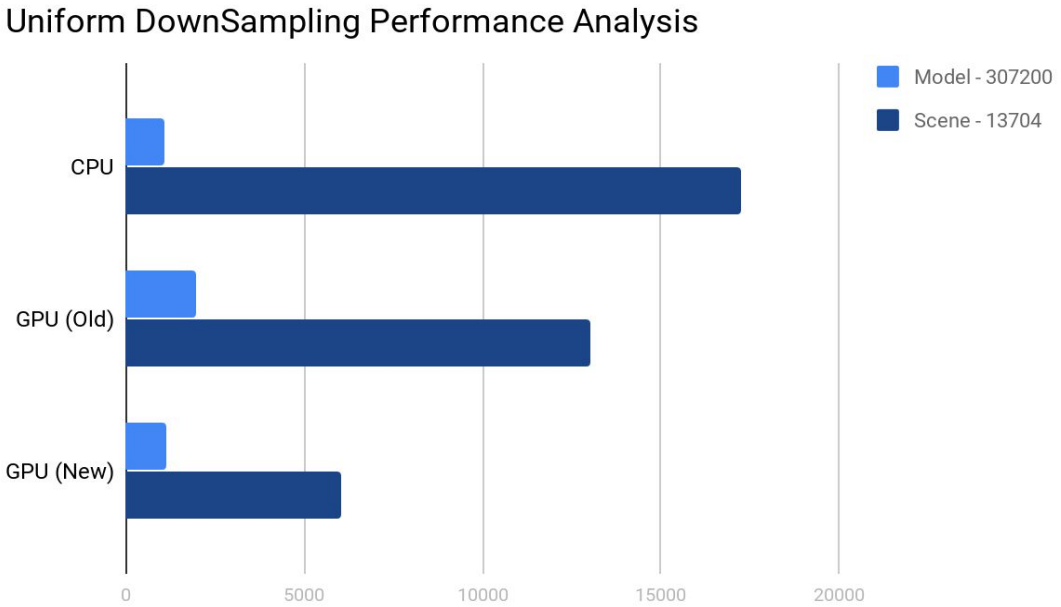
Original

- Scene Min, Max Calculation with CUDA
- Divide scene into grids based on *radius,* compute grid indices and distances to grid center
- Sort the grid indices
- Find unique indices
- Use CPU to sort within distance within each grid

New

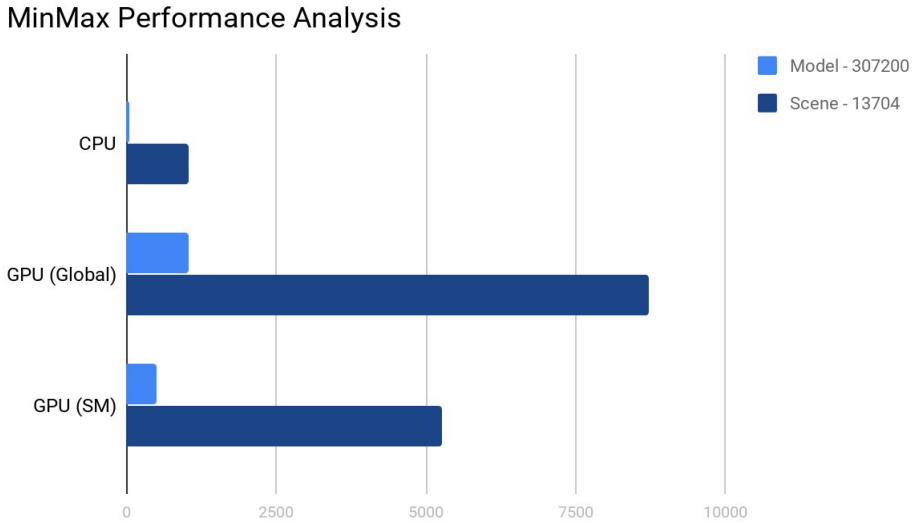
- Scene Min, Max with CPU
- Divide scene into grids based on *radius,* compute grid indices and distances to grid center. Use AtomicMin and global memory to store the min distances of each grid
- Check if the point has distance = the min distance of the grid that it belongs to
- Save its index if yes

Performance Gain



CPU	1085	17216
GPU (Old)	1977	12995
GPU (New)	1129	6021

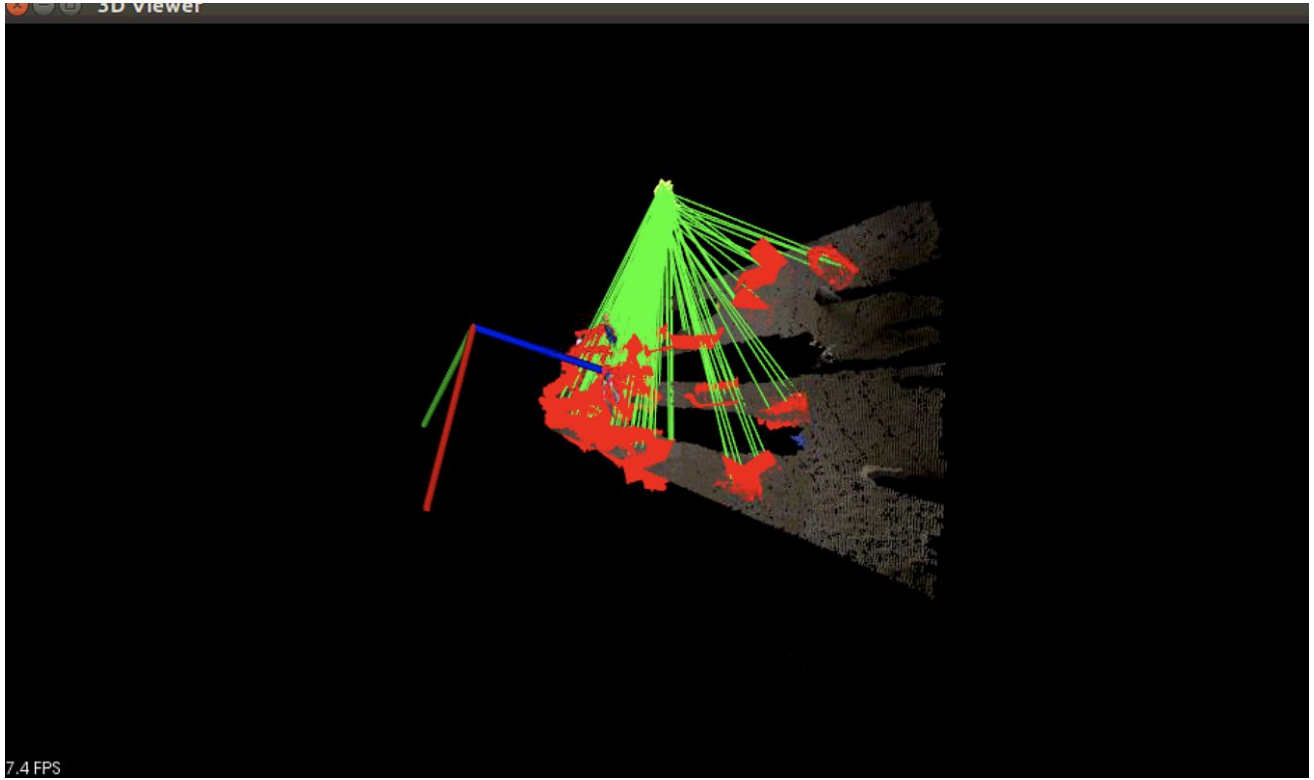
Why CPU for MinMax Calculation?

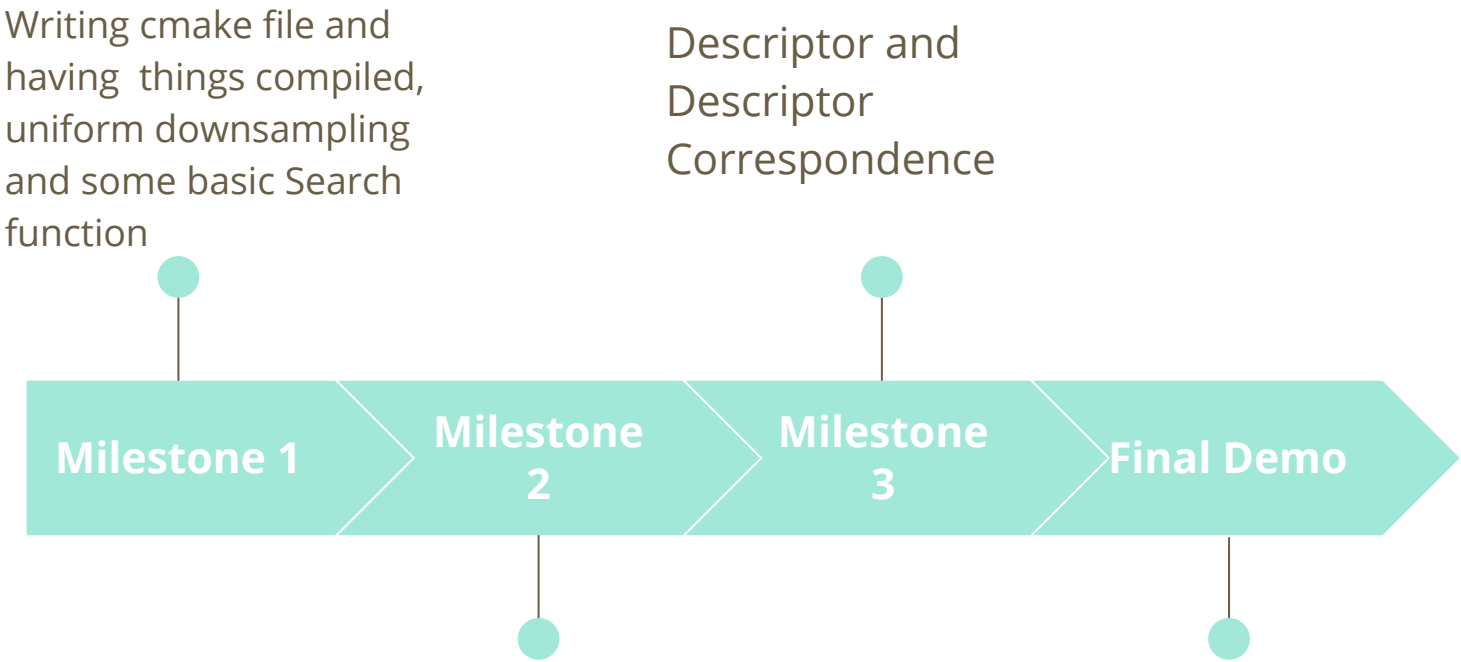


CPU	60	1035
GPU (Global)	1055	8712
GPU (SM)	500	5258

```
__global__ void getMinMax(int N ,const PointType *pts_in, Eigen::Vector4f *min_pt, Eigen::Vector4f  *max_pt){
    __shared__ float min_max[6];
    for (int i = 0; i < 3; ++i)
        min_max[i] = FLT_MAX;
    for (int i = 3; i < 6; ++i)
        min_max[i] = FLT_MIN;
    __syncthreads();
    int index = threadIdx.x + (blockIdx.x * blockDim.x);
    if (index < N){
        PointType pt = pts_in[index] ;
        if (isfinite(pt.x) && isfinite(pt.y) && isfinite(pt.z)){
            atomicMin(&min_max[0], pt.x);
            atomicMin(&min_max[1], pt.y);
            atomicMin(&min_max[2], pt.z);
            atomicMax(&min_max[3], pt.x);
            atomicMax(&min_max[4], pt.y);
            atomicMax(&min_max[5], pt.z);
        }
    }
    __syncthreads();
    atomicMin(&(*min_pt)[0], min_max[0]);
    atomicMin(&(*min_pt)[1], min_max[1]);
    atomicMin(&(*min_pt)[2], min_max[2]);
    atomicMax(&(*max_pt)[0], min_max[3]);
    atomicMax(&(*max_pt)[1], min_max[4]);
    atomicMax(&(*max_pt)[2], min_max[5]);
}
```

N_ after sampling must be smaller than original point cloud size! But with improper radius setting, this wouldn't be true.




Writing cmake file and having things compiled, uniform downsampling and some basic Search function
Descriptor and Descriptor Correspondence
Milestone 1
Milestone 2
Milestone 3
Final Demo
Fixing multiple errors, working on descriptors and search methods
Grouping and transformation
Possible Optimization

Next steps

Descriptor

Finish interpolation

All the Rest, Analysis and Optimization