## Implementation and Results

The solution was built using C# 4.0, DirectX 9, shader model 3.0, without compute shader). It was design to be easily been added to an existing application. The only considerations are fist to execute the occlusion optimization routines, and second to add the visibility check in the main vertex shader of the application.

A 3D city model was built, composed of 210 meshes, adding up a total of 379.664 triangles. For this scene 258 Occluders were generated in Offline time by the process detailed in [PAPER\_MATIAS]. In order to analyze the algorithm performance, fifteen representative scene View Points were taken. For each position we compute the following metric: Value = (t - v) / t \* 100, where t is the total scene meshes and v is the total visible meshes.

This metric allows us to see the percent of discarded meshes that Occlusion Culling prevented from sending to the GPU in each frame. The metric is computed with Occlusion Culling deactivated and then with it activated. We also include the frames per second that resulted from rendering the scene with and without Occlusion Culling. The results were computed using a PC with Intel Core i3 2.40GHz processor with 2GB RAM and Intel HD Graphics 3000 GPU.

Fig. XXXXXXXXX. Top: FPS rendering performance only with Frustum Culling and then with Occlusion Culling activated, at the fifteen different selected View Points. Bottom: Discarded mesh percent, first with only Frustum Culling and then activating Occlusion Culling, at the fifteen different selected View Points.