



Dynamic Environment Mapping for Augmented Reality Applications on Mobile Devices —Supplemental Material—

R. Monroy , M. Hudon and A. Smolic 

V-SENSE, School of Computer Science and Statistics
 Trinity College Dublin, Dublin, Ireland



Figure 1: Sample results obtained with our pipeline. *Left to right*: Resulting rendering of the Stanford Dragon after acquiring an office-space environment, capture of the mobile device from the user’s point of view showing the Stanford Bunny, frame as seen directly on the device.

This document contains additional material for the paper "Dynamic Environment Mapping for Augmented Reality Applications on Mobile Devices". Details on the GPU implementation of the colour correction matrix and SH coefficients calculation are provided. Additionally, visual comparisons of the resulting translated EMs are presented.

1. GPU Implementation

The OpenGL Shading Language defines the concept of compute shaders, which can be used to perform any sort of parallel computation not related to the rendering pipeline. When a compute shader is dispatched, it creates one or several *work groups*, which contain a collection of shader invocations, referred to as *local invocations*. Within a *work group*, it is possible to define variables allocated in *shared memory*, which is accessible only to invocations belonging to the same *work group*. Read and write operations on this kind of memory are faster than in global memory.

The calculation of the colour correction matrix and the SH coefficients involve a series of operations whose result needs to be summed. This can be treated as prefix sums, which are highly parallelisable [Ble90]. For both, the correction matrix and SH coefficient computation, the shader is dispatched twice. In the first pass, several *work groups* are spawned, each with many *local invocations*. These *local invocations* operate over a section of the input data (global memory) and accumulate it into its reserved segment in the *shared memory*. The invocations are then synchronised using a *memory barrier* that ensures all write operations to *shared memory* have finished. The last step in the first pass is restricted to only

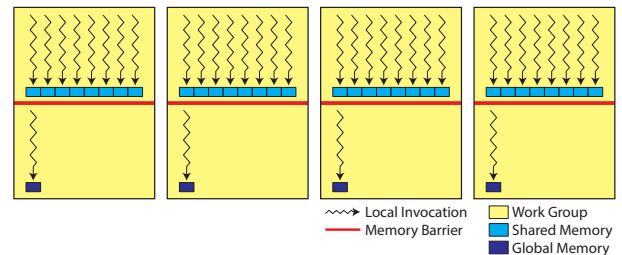


Figure 2: First pass in prefix sum.

one invocation per *work group* that is used to accumulate all the values in the *shared memory* and stores them in its allocated slot in global memory. Fig. 2 illustrates the first pass. For brevity only four *work groups* are shown, each with eight *local invocations*. In practice, this number is considerably higher. In the second pass, only one *work group* is spawned with one *local invocation* that accumulates the results stored in global memory and its result is also stored in global memory.

2. EM Translation Comparisons

A visual comparison of the effects when the EM is translated is shown in Fig. 3. The corresponding Data Completeness (DC) and Correlation Coefficient (CC) are shown at the top of each image.

References

[Ble90] BLELLOCH G. E.: *Prefix Sums and Their Applications*. Tech. rep., Synthesis of Parallel Algorithms, 1990. 1

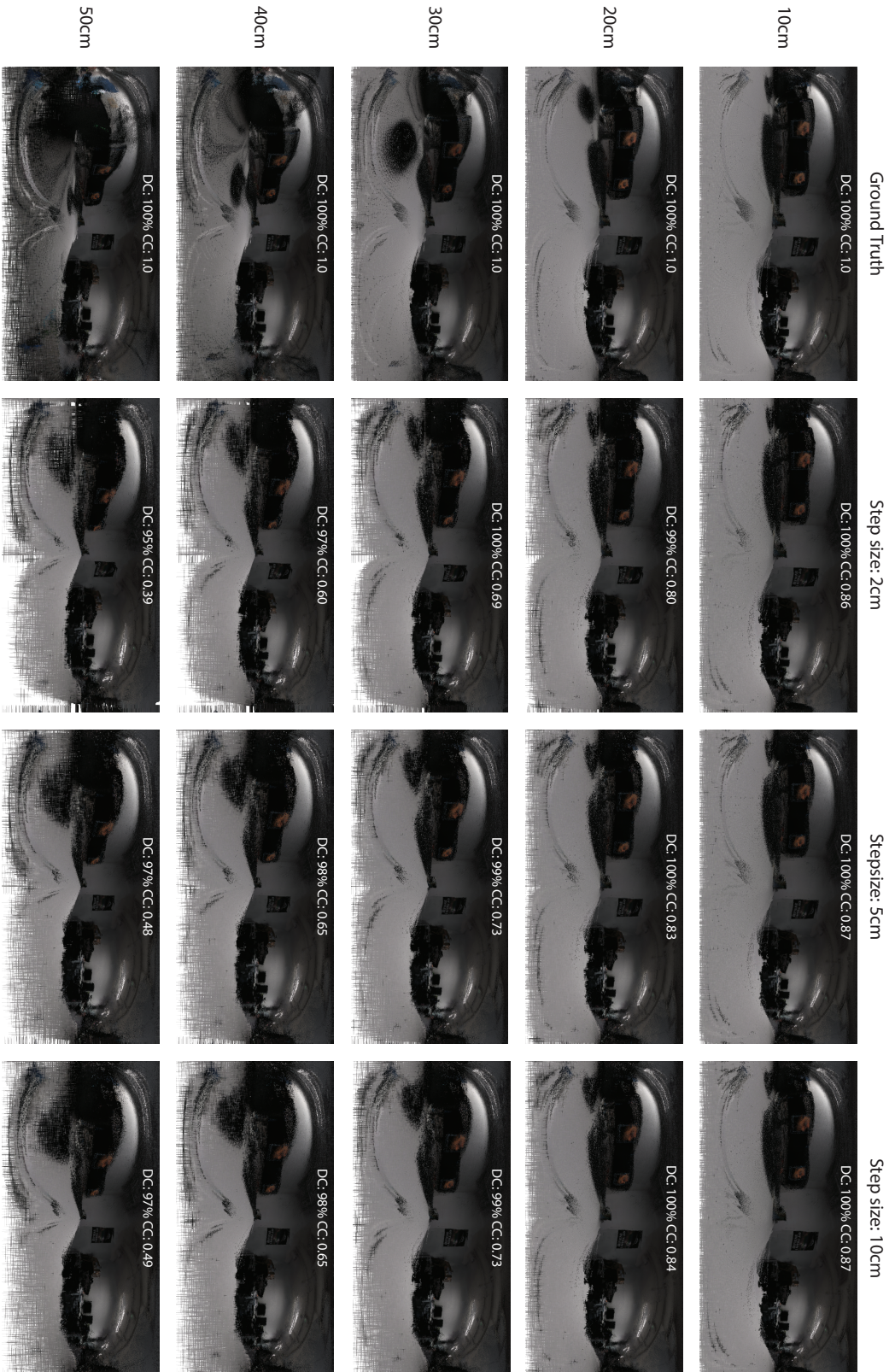


Figure 3: Visual comparison when continuously translating an EM 50cm along the x-axis in steps of 2cm, 5cm and 10cm. Columns contain the EM translations. The Ground Truth (shown in the first column) is estimated by projecting the RGB-D data to an EM with an origin matching that of the translated EM. The cross pattern observed in some of the images, particularly in regions close to the nadir, corresponds to the search algorithm mentioned in Section 7.1 when calculating the SH coefficients.