VISION - Practical work 4

Leo PORTE

November 5, 2023

Firstly, let's play with lambda's value. Results with default λ value (with $\lambda f = 0.1$) are as shown below:

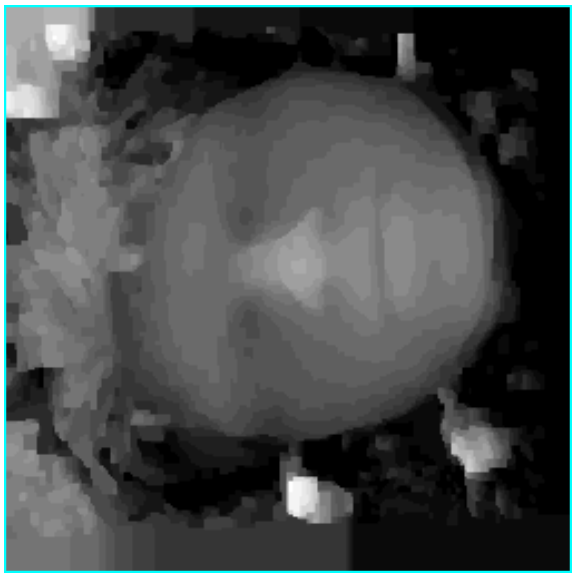

Figure 1: Default disparity map

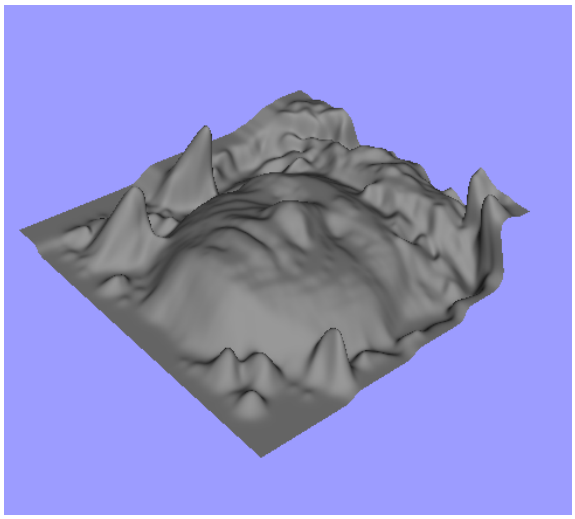

Figure 2: Default depth map after blurring on disparity map

Now let's put λf=0.35:

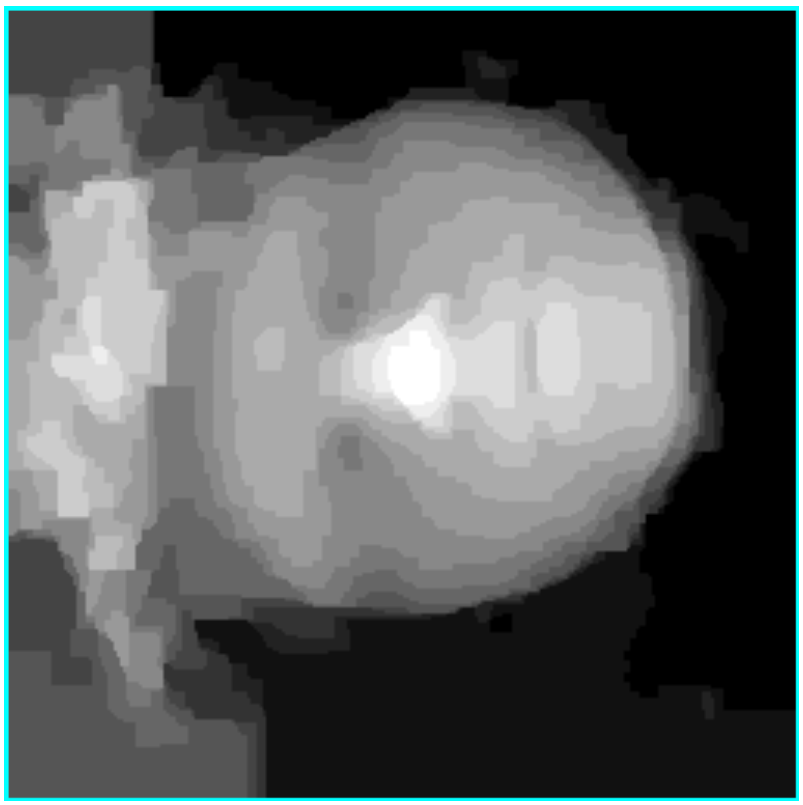

Figure 3: Disparity map with $\lambda f = 0.35$

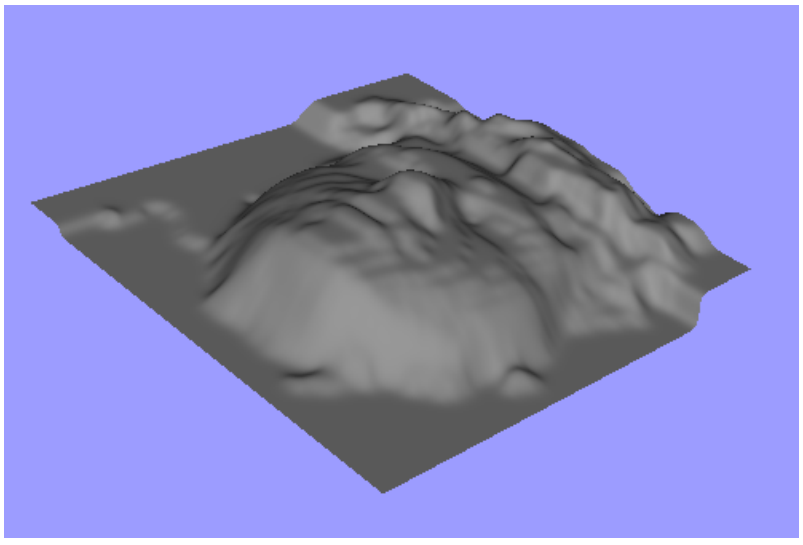

Figure 4: Depth map with $\lambda f = 0.35$ after blurring on disparity map

And now λf=2:

Figure 5: Disparity map with $\lambda f = 2$

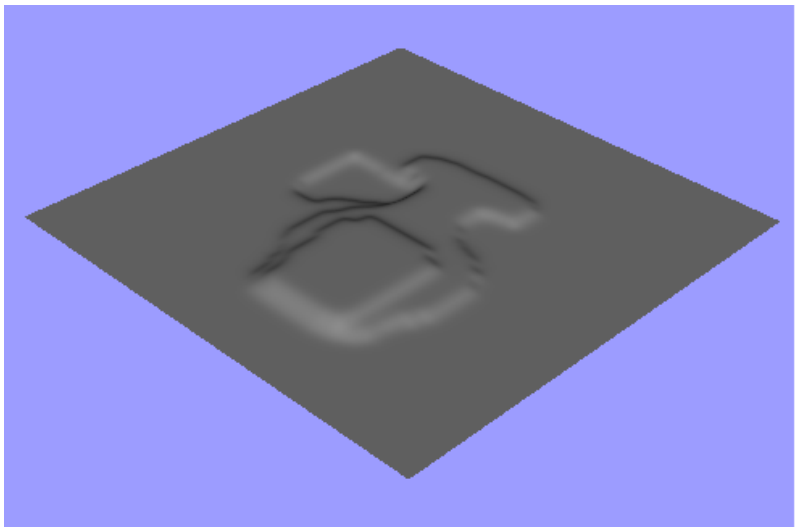

Figure 6: Depth map with $\lambda f = 2$ after blurring on disparity map

As long as we increase the value of λf, our algorithm seems to be less and less sensitive to relief changes. Moreover, it seems to lower the number of relief changes which are possible to capture (easily more than 10 for small λf values and only 4 for $\lambda f = 2$.

As our default result was too noisy, increasing λf value permitted to get better results, having a depth map which covers essentially the face. For the rest of the report, we will keep $\lambda f = 0.35$.

We can see that our disparity maps are a little bit pixelised before blur processing. Hence, we suppose that a median filter used after the graph decoding could help us in reducing this aspect and having an even smoother depth map at the end. Let's see if it is true:

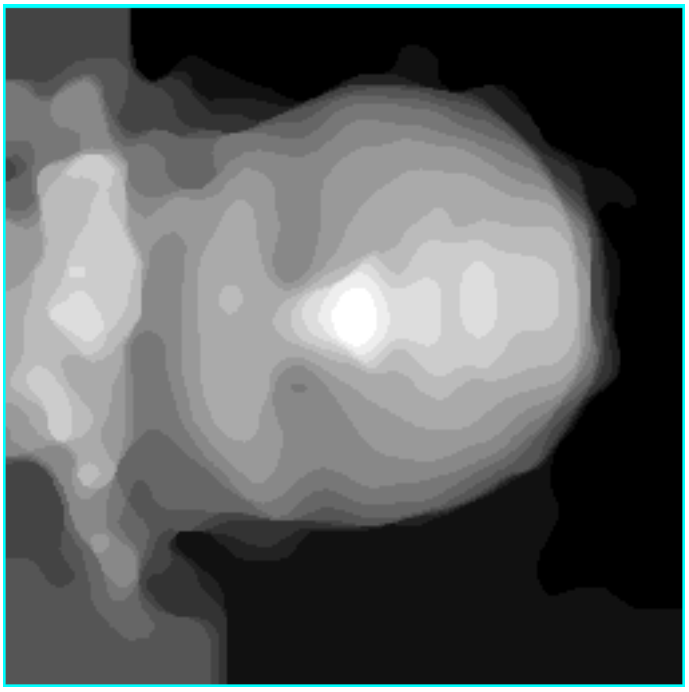

Figure 7: Disparity map with size of filter = 10

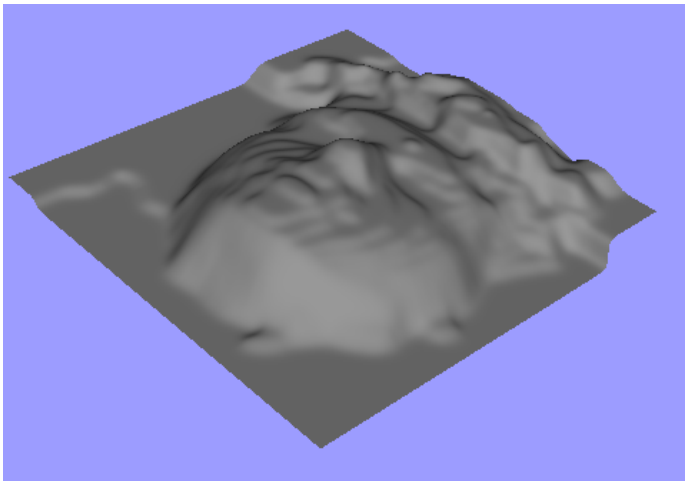

Figure 8: Depth map with size of filter = 10

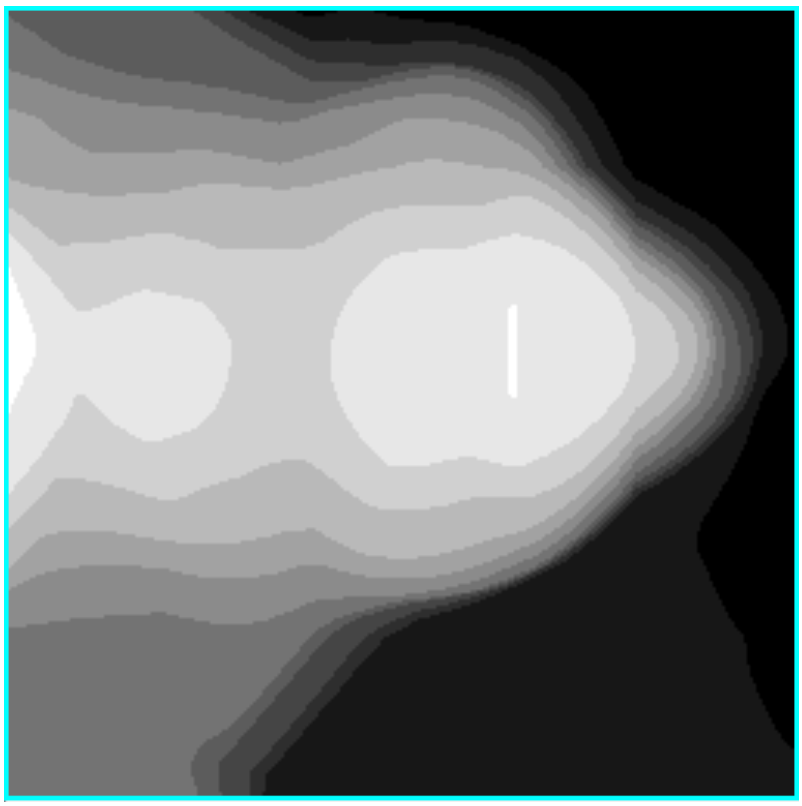

Figure 9: Disparity map with size of filter = 100

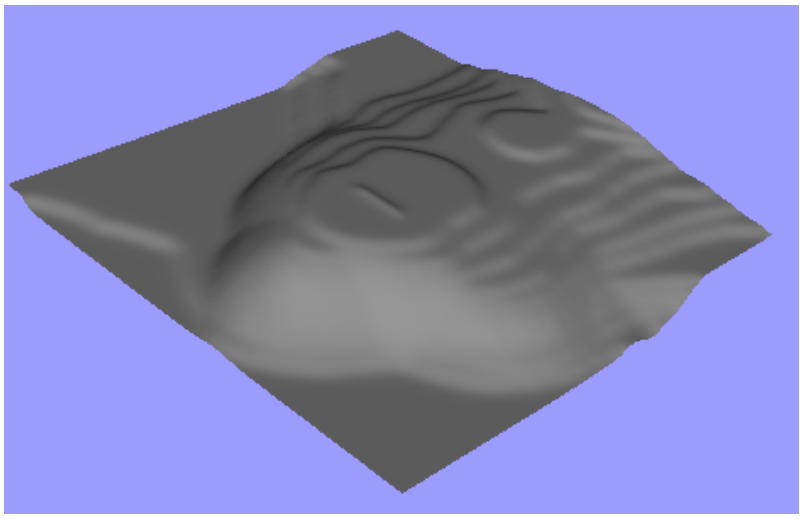

Figure 10: Depth map with size of filter = 100

Applying a median filter after having decoded the graph seems to smooth even more final result. However, changes are too subtle or negligent if filter size is too small, and we loose too much details if we increase it. Standard blur is sufficient in our case.

Now, we will toggle values attributed to D(x,y) in the decode_graph function.

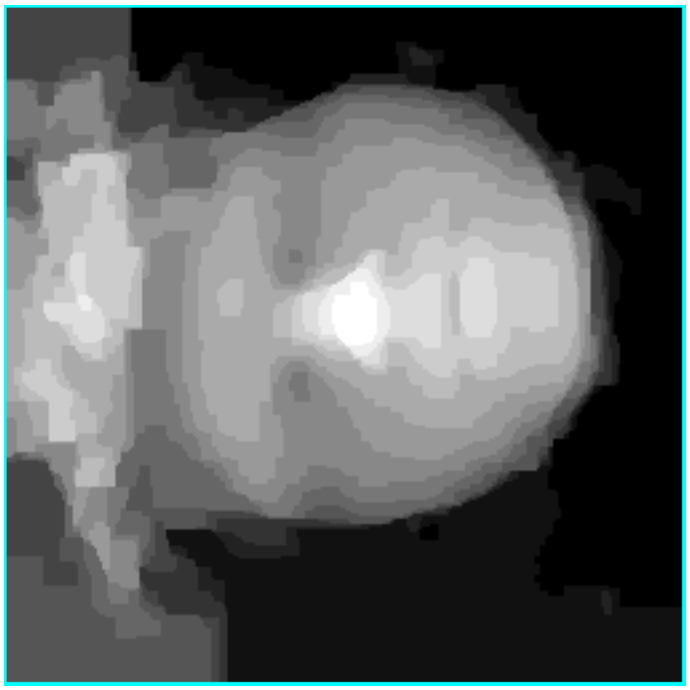

Figure 11: Disparity map with D(x,y) $\times$ 5

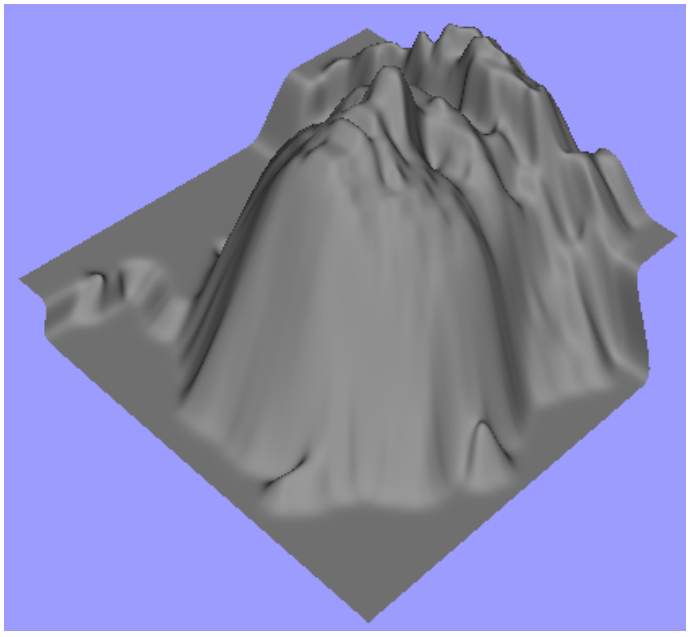

Figure 12: Disparity map with D(x,y) $\times$ 5

We can see that toggling this number doesn't change disparity map (obviously), but it changes the depth of the layers in the depth map, by increasing it.

Finally, after all these experiments, we can define a convenient final result, as shown below:

Figure 13: Final result

Comparison between GC Disparity and region-growing

The comparison is based on three main criteria: precision, smoothness, and computation time.

Precision

- **Graph Cuts:** Known for high precision, especially in textured and well-defined areas. Global optimization ensures consistency with image data and smoothness constraints.
- **Region-Growing:** Precision depends on seed quality and expansion criteria. May struggle with textureless or repetitive patterns.

Smoothness

- **Graph Cuts:** Incorporates smoothness constraints, leading to smooth disparity maps that preserve edges. Controlled by the smoothness parameter λ.
- **Region-Growing:** Smoothness varies and is not guaranteed. The method may produce patchy results without additional processing.

Computation Time

- **Graph Cuts:** Computationally intensive due to global optimization. Optimizations can reduce processing time (in our case using a zoom).
- **Region-Growing:** Potentially faster, especially with limited seed points and simple criteria. Time complexity can increase with the number of regions.

Other Considerations

- **Robustness to Noise:** Graph cuts are more robust to noise, while region-growing can propagate errors from noisy seeds.

- **Parameter Tuning:** Both methods require parameter tuning, which is crucial for performance.
- **Handling Occlusions:** Graph cuts can explicitly handle occlusions, whereas region-growing may need additional steps.
- **Scalability:** Graph cuts are more memory-intensive, which can be a concern for large-scale problems. Region-growing can be more scalable with efficient implementation.