Lucas Versini

Practical Session n°4
October 26, 2024

3D Computer Vision

Master MVA

Practical Session n°4

Section 1 briefly explains the objective of this practical session.
Section 2 explains how the program works. This is to expose my understanding of the session, and to explain some parts of the code.
Section 3 presents some results, which are compared with the seeds expansion method.

1 Objective

This section is the same as in the previous Practical Session.

We are given two rectified images, I_L and I_R, of the same scene. Rectification implies that the epipoles are at infinity or, practically, that the cameras are simply translated between the two views.

The goal is to compute the disparity map for the left image I_L. The disparity d corresponds to the horizontal shift between corresponding pixels in I_L and I_R. From this, we can infer the depth z of objects in the scene: closer objects exhibit larger disparities, while objects farther away exhibit smaller ones: $z \propto \dfrac{1}{d}$.

2 Technical explanation

We estimate disparities using a graph cut. Specifically, if the possible values for the disparity are $d_{\text{min}}, \dots, d_{\text{max}}$, then for each pixel p we create $d_{\text{max}} - d_{\text{min}}$ nodes $p_1, \dots, p_{d_{\text{max}} - d_{\text{min}}}$.
We also add a source node s, and a target (or sink) node t.

The weights we use are shown in Figure 1, where p and q are neighboring pixels, and where $w_j^p = D_p(j) + K_p$, with $D_p(j) = w_{cc} \min\left(1, E_{ZNCC}\left(P; (j, 0)\right)\right)$, and $K_p = 1 + (d_{\text{max}} - d_{\text{min}})|N_p|\lambda$, with $|N_p|$ the number of neighbors of p ($d_{\text{max}} - d_{\text{min}}$ comes from the fact that the possible disparities are $d_{\text{max}}, \dots, d_{\text{min}}$, hence $d_{\text{max}} - d_{\text{min}} + 1$ possible values).

Note that $D_p(j)$ essentially measures how plausible it is to have disparity j for pixel p, while K_p ensures that an optimal cut in the graph will only cut once through the line of nodes corresponding to p.

Depending on the pixel p (inside the image, on an edge, in a corner), we have $|D_p| = 2$ (corner of the image), $|D_p| = 3$ (edge of the image) or $|D_p| = 4$ (inside the image). So in the code, we compute the true number of neighbors.
However, for a fixed p, increasing K_p to a larger value is not a problem: indeed, K_p guarantees that the line of nodes corresponding to p is only cut once, so increasing all the weights for p of a same amount will not change this fact, it will simply add a constant to the value of the optimal cut. So in fact, we could just use the value 4 for each pixel, this would not change the optimal cut. We verified this with the image from next section, by checking that using the real

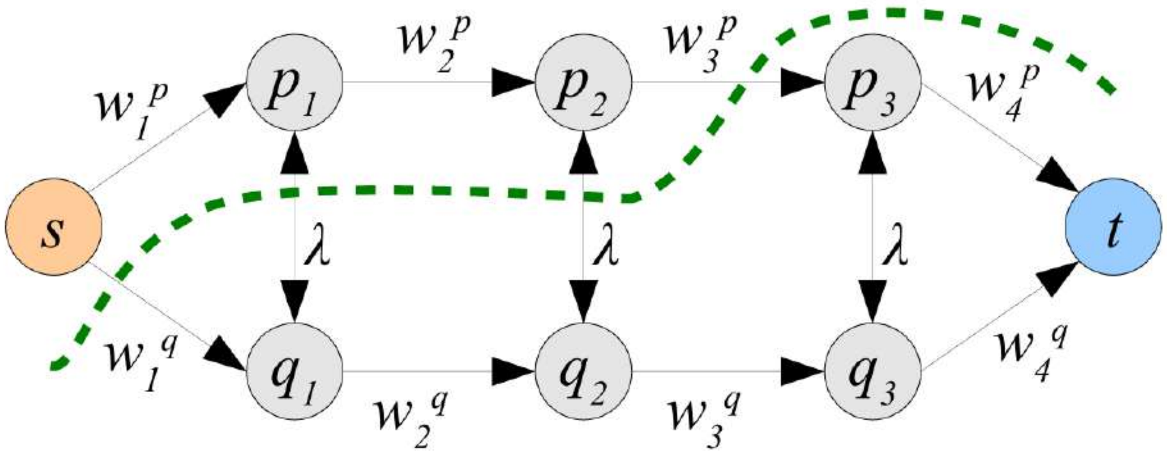


Figure 1: Graph for disparity estimation (taken from the slides)

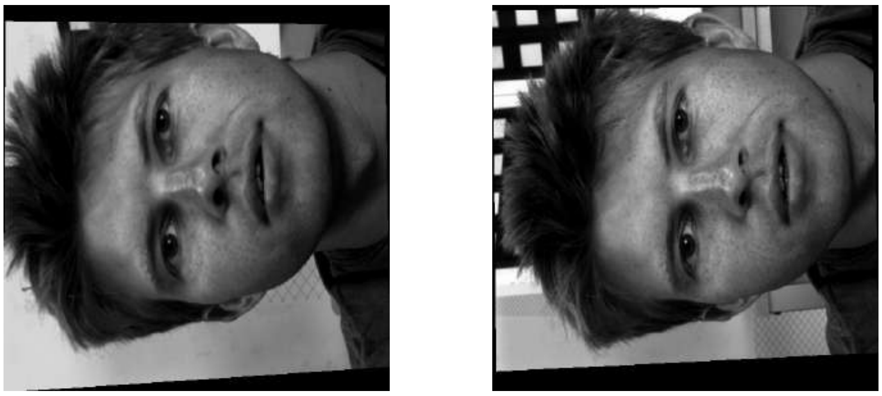
number of neighbors or 100 did not change the final graph cut.

A last technical point we detail is the fact that for each node (x, y, d), we need to associate an index (a single integer). We use the formula $\texttt{d} + \texttt{nd} * (\texttt{y} + \texttt{x} * \texttt{ny})$ (line 202 of the code), though of course, we could exchange the variables.

3 Results

3.1 Basic results

We work with the images shown in Figure 2.

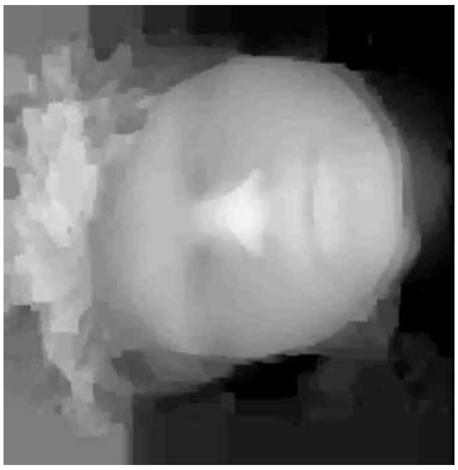


(a) First image (b) Second image

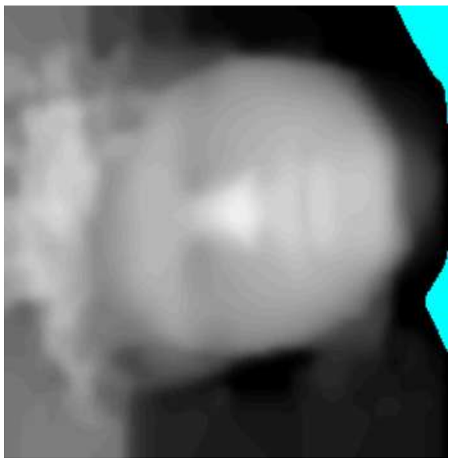
Figure 2: Images used for disparity estimation

The results obtained with the graph-cut method can be seen in Figure 3.

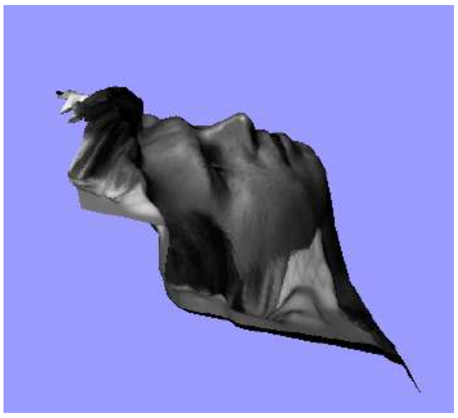
These results are quite impressive: the initial disparity map is already good, but blurring it erased some noise, while preserving the edges. In the end, the 3D reconstruction looks really good.
We see, for the blurred disparity map, a lot of blue on the right. In the non-blurred disparity map, there is a line of blue (simply because pixels near edges are "not valid", in the sense that we will not have valid patches, hence no disparity), which is amplified when blurring the image.



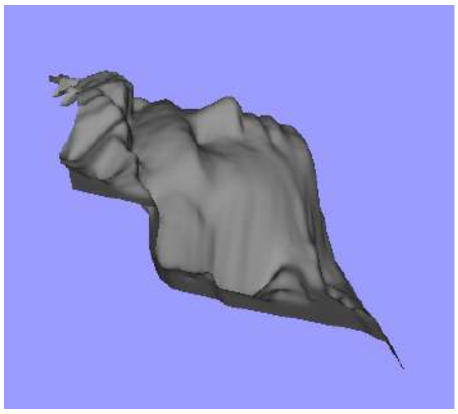
(a) Disparity map (graph cut)



(b) Blurred disparity map (graph cut)



(c) 3D estimation (graph cut)



(d) Blurred disparity map (graph cut)

Figure 3: Results of the graph cut

3.2 Impact of the parameters

When not specified otherwise, we will use $\lambda = 0.1$ and `win` $= (7 - 1)/2$.

3.2.1 Impact of λ (`lambdaf`)

Here, we discuss the impact of the parameter λ.
Intuitively, the larger λ is, the more costly it is to have neighbors with different disparities. So we expect the disparity map to be really smooth for large values of λ (so that neighbors typically have the same disparity), and noisy for small values of λ (because then, having neighbors with different disparities is not really costly).

We compare the results for λ equal to 0.01, 0.1 and 0.5.
The results are visible in Figure 4.

They are coherent with what we anticipated above.

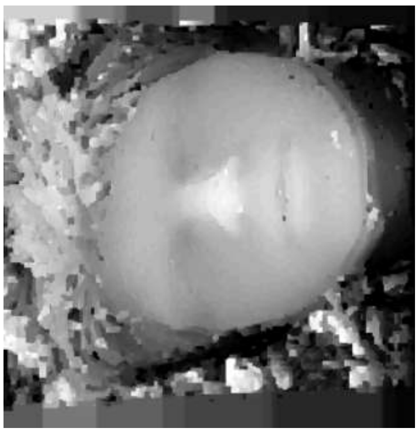
3.2.2 Impact of the window size

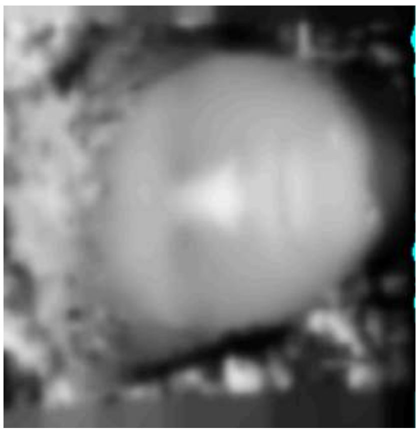
Here, we discuss the impact of the parameter `win`, which defines the size of the considered patches.
We compare the results for `win` equal to $(5 - 1)/2$, $(7 - 1)/2$ and $(15 - 1)/2$.
The results are visible in Figure 5.

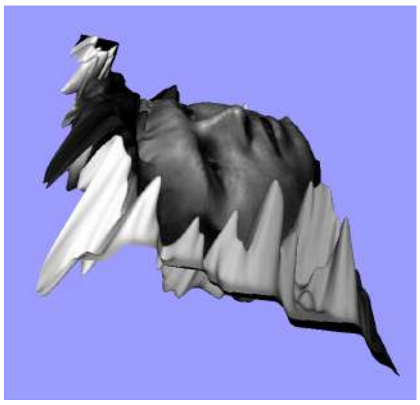
We do not notice much of a difference.

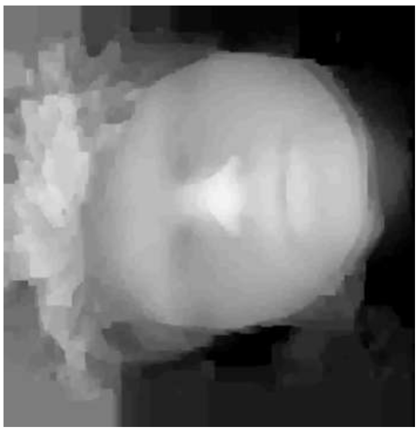
A larger value of `win` leads to more computation time.

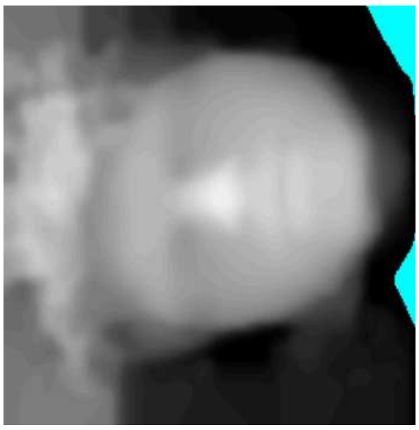
Theoretically, though once again it is here to observe here, we expect a large value of `win` to lead to a smoother image (because there will be less noise, using a larger window), and a small value of `win` to give better results for the details while being noisy (because then, essentially just the pixel of interest is used, not the context formed by its surroundings).

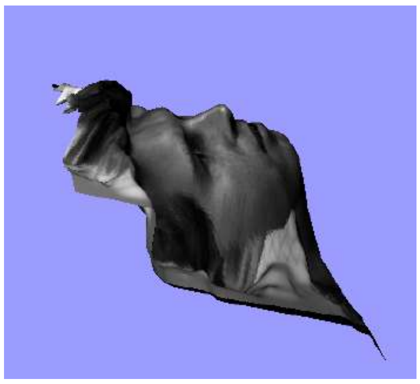

(a) Disparity map, $\lambda = 0.01$

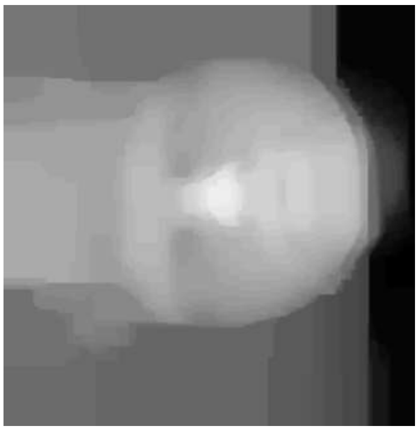

(b) Blurred disparity map, $\lambda = 0.01$

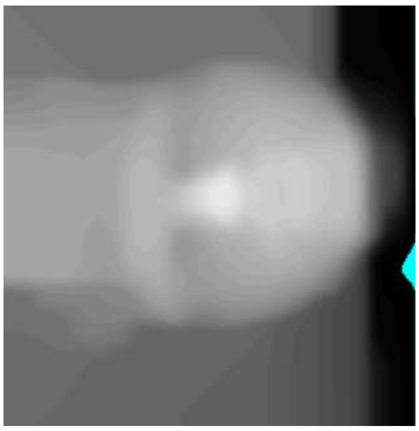

(c) 3D estimation, $\lambda = 0.01$


(d) Disparity map, $\lambda = 0.1$


(e) Blurred disparity map, $\lambda = 0.1$


(f) 3D estimation, $\lambda = 0.1$


(g) Disparity map, $\lambda = 0.5$


(h) Blurred disparity map, $\lambda = 0.5$

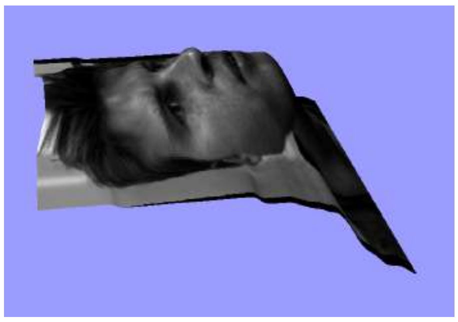
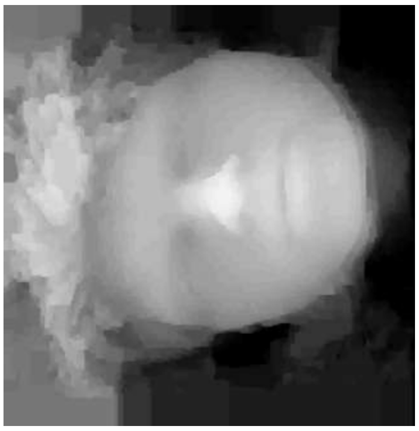
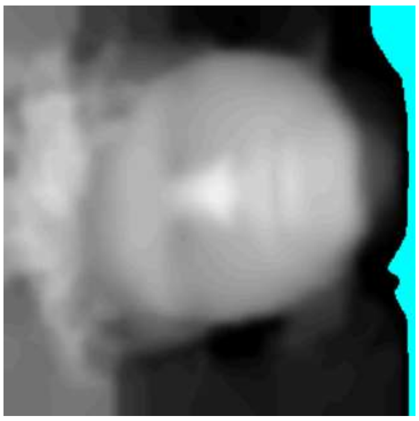
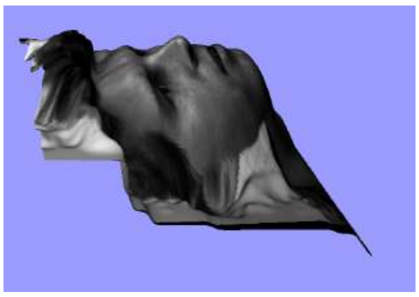
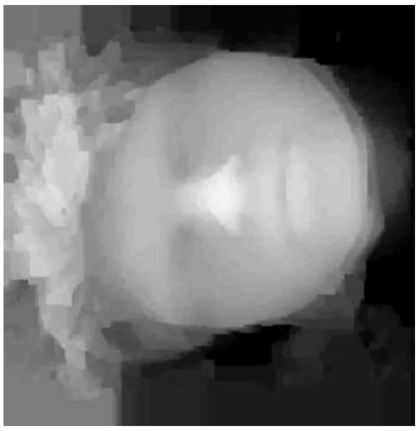

(i) 3D estimation, $\lambda = 0.5$

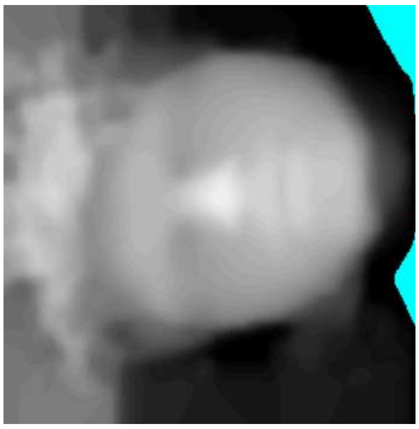
Figure 4: Impact of the parameter λ

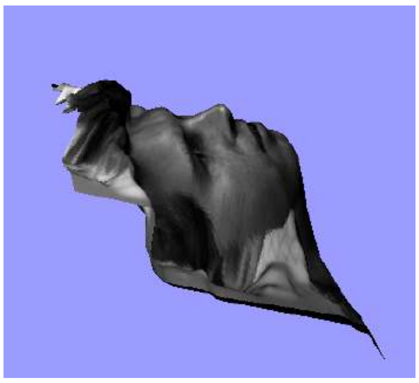

(a) Disparity map, `win` $= (5-1)/2$

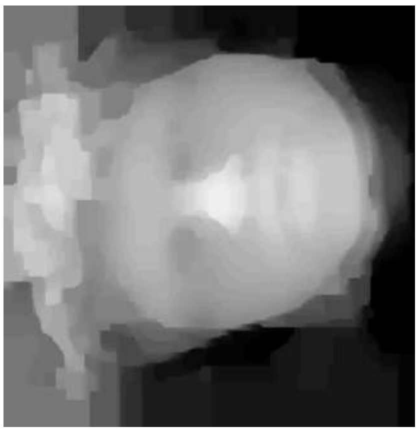

(b) Blurred disparity map, `win` $= (5-1)/2$

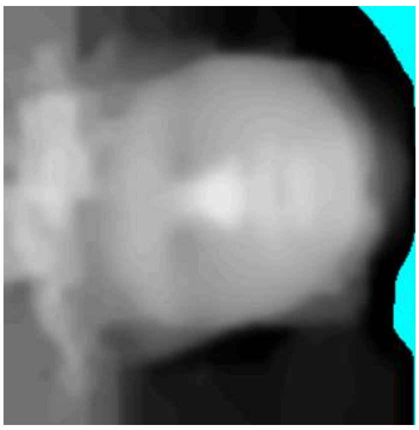

(c) 3D estimation, `win` $= (5-1)/2$


(d) Disparity map, `win` $= (7-1)/2$


(e) Blurred disparity map, `win` $= (7-1)/2$


(f) 3D estimation, `win` $= (7-1)/2$


(g) Disparity map, `win` $= (15-1)/2$


(h) Blurred disparity map, `win` $= (15-1)/2$

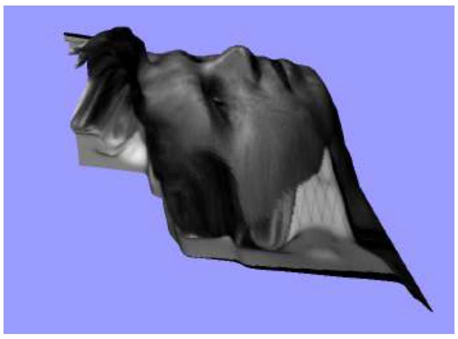
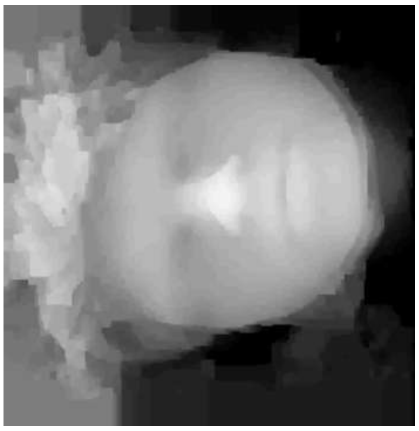
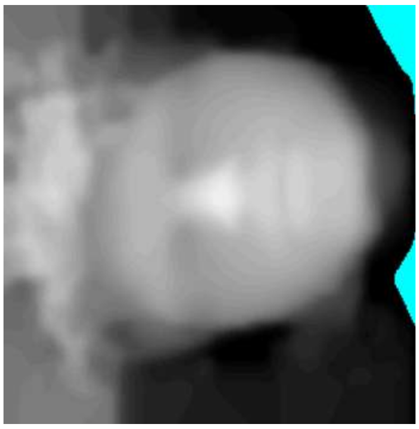

(i) 3D estimation, `win` $= (15-1)/2$

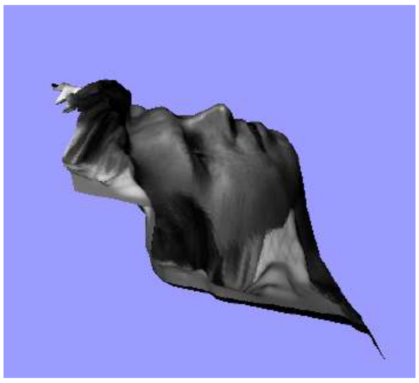
Figure 5: Impact of the parameter `win`

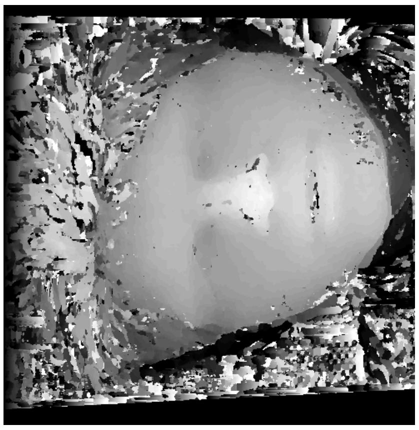
3.3 Comparison with the seeds expansion method

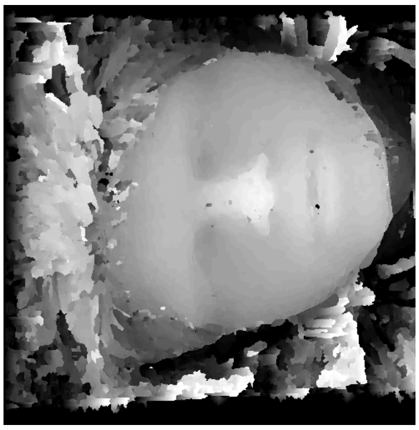
We compare the results of the graph-cut algorithm with the seeds expansion-algorithm, seen in the previous practical session.
The results can be seen in Figure 6.


(a) Disparity map, graph-cut


(b) Blurred disparity map, graph-cut


(c) 3D estimation, graph-cut


(d) Dense disparity map


(e) Disparity map, seeds expansion

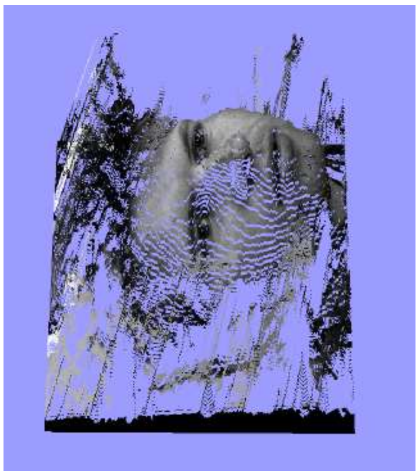

(f) 3D estimation, seeds expansion

Figure 6: Comparison graph-cut / seeds expansion (first image)

We observe that the disparity map is noisy for the seeds expansion method, while the one obtained with graph-cut, even without blurring, is not.

As for the 3D reconstructions, the one obtained with graph-cut is smooth (as observed before), while the one obtained with seeds expansion is not that good: though we do see volume, we notice a lot of noise. This is because seeds expansion does not really use regularization, or at least not like graph-cut: the pixels next to seeds with estimated disparity d must have an estimated disparity of $d-1$, d or $d+1$, but obviously this is not enough to get a smooth result.

And on the laptop used to run the above experiments, seeds expansion took about twice as long as the graph-cut method.