Lucas Versini

Practical Session n°3
October 20, 2024

3D Computer Vision

Master MVA

Practical Session n°3

Section 1 briefly explains the objective of this practical session.
Section 2 explains how the program works. This is simply to expose my understanding of the session, which helps me, but you can definitely skip it.
Section 3 presents some results, and some limitations.

1 Objective

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: $z \propto \frac{1}{d}$.

2 Technical explanation

Given a pixel $p = (x, y)$ in I_L, we seek to find its disparity d, such that p in I_L matches with $(x + d, y)$ in I_R. This can be achieved by maximizing the similarity between patches in both images, using Normalized Cross-Correlation (NCC).

To do this, we define a square window W, where $W = [-\mathtt{win}, \mathtt{win}]^2$. We then compute the NCC between the patch centered at p in I_L and the patch centered at $p + d$ in I_R, using the formula:

$$d(p) = \arg\max_d \frac{\sum_{r \in W} \left(I_L(p+r) - \overline{I}_L^{W(p)}\right)\left(I_R(p+r+de_1) - \overline{I}_R^{W(p+de_1)}\right)}{\sqrt{\sum_{r \in W} \left(I_L(p+r) - \overline{I}_L^{W(p)}\right)^2 \sum_{r \in W} \left(I_R(p+r+de_1) - \overline{I}_R^{W(p+de_1)}\right)^2}}, \tag{1}$$

where $\overline{I}_L^{W(p)}$ and $\overline{I}_R^{W(p+de_1)}$ are the average values of the intensities in the patches W centered at p in I_L and at $p + d$ in I_R, respectively.

The full process is as follows:

- For each pixel (x, y) in I_L, we compute the NCC for all disparity values d between two limit values $d_{\min}$ and $d_{\max}$. We retain the disparity value $d(p)$ that maximizes the NCC, provided that the maximum NCC exceeds a certain threshold. Pixels for which the NCC is above a fixed threshold are marked as "seeds", and stored in a priority queue, ordered by the NCC value.

- For each seed pixel p, we explore its immediate neighbors (left, right, top, bottom). For the neighbors which are not seeds, we compute their disparity by restricting d to the values $d(p) - 1$, $d(p)$, and $d(p) + 1$ in (1), and add them to the queue. This is why this approach is called *seed expansion*.
- Finally, we can visualize the 3D reconstruction of the scene using the disparity map.

3 Results

3.1 Basic results

In this section, we present some results for the disparity computation and the corresponding 3D reconstruction.

Figure 1 shows the results for the images given in the practical session.

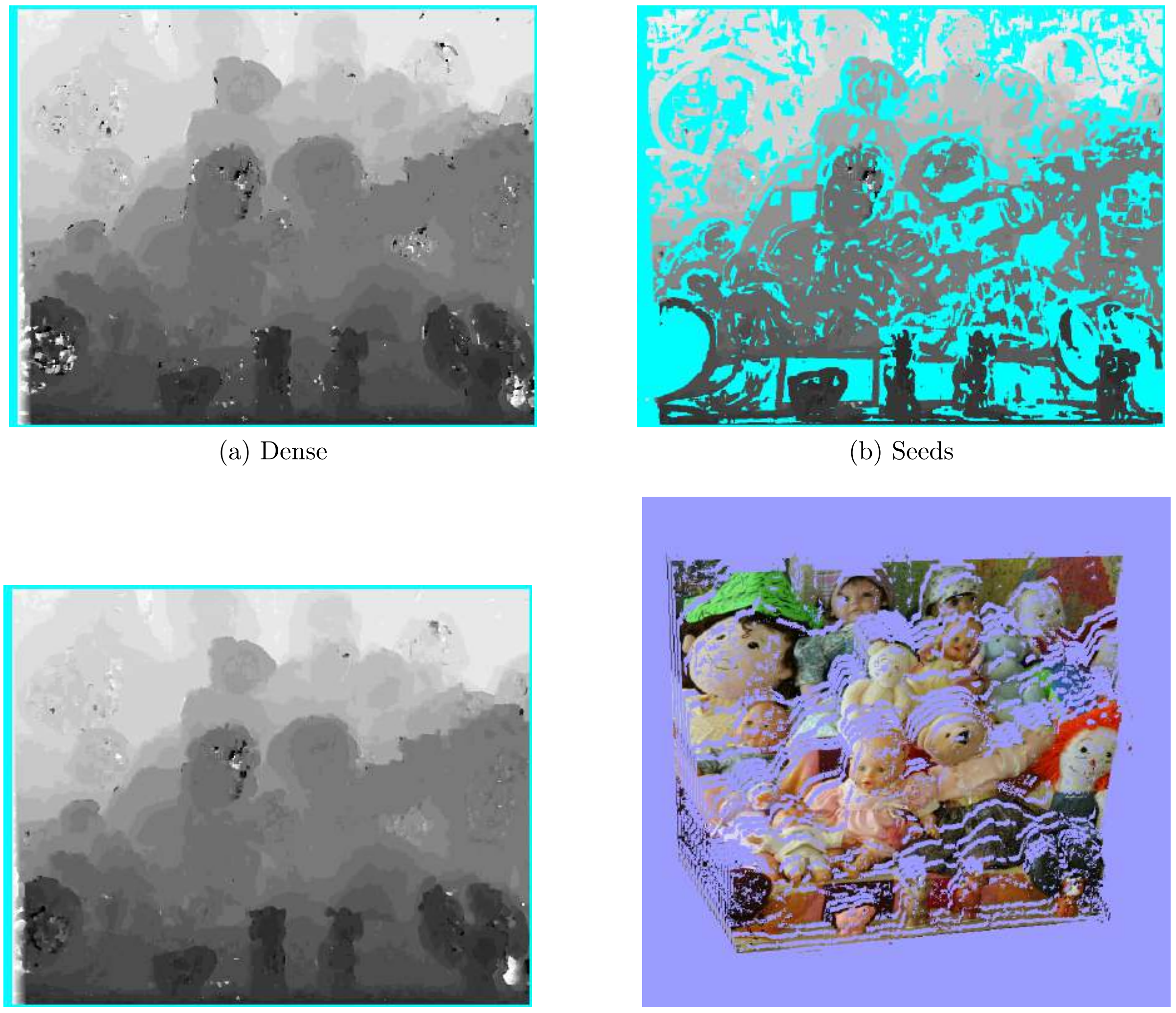

(a) Dense (b) Seeds (c) Propagation (d) 3D

Figure 1: Experiments on the given images

- Figure (a) shows the disparity map obtained without using seed expansion, that is, for each pixel p, we simply compute $d(p)$ as in (1).

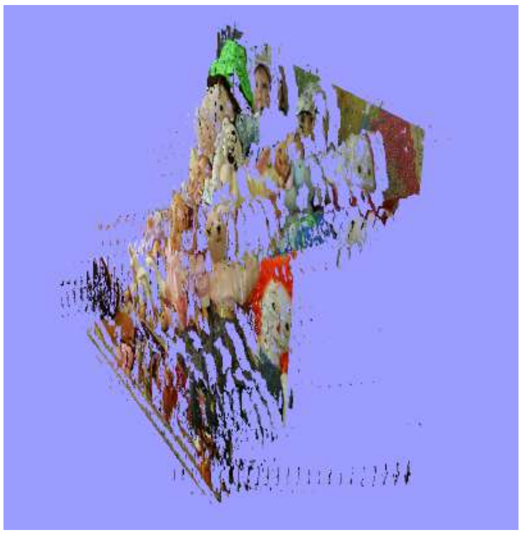


Figure 2: Some errors in the 3D reconstruction

- Figure (b) shows the disparity map for the seeds only: the pixels in blue are the ones that are not seeds.

  On the left part of the image, an object is entirely in blue. This is probably because this area is rather uniform, so the NCC is close to 0, see Subsection 3.2.

- Figure (c) shows the final disparity map after applying seed expansion.

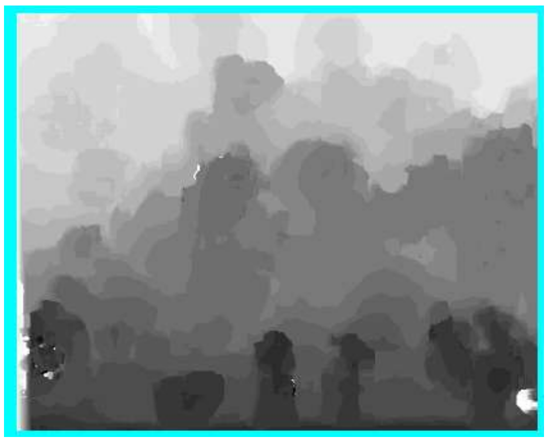
- Figure (d) shows the 3D reconstruction of the scene.

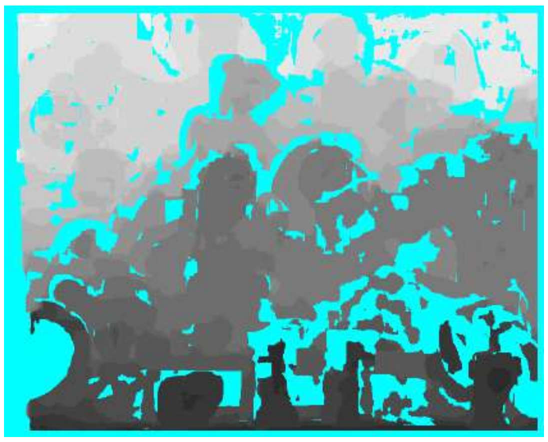
We see that between (a) and (c), things are smoother: the edges, but also some artifacts that mostly appeared in white in (a). That is what seed expansion does: by constraining $d(p')$ to be $d(p) - 1, d(p)$ or $d(p + 1)$ for p' a neighbor of the seed p, some regularity (or smoothness) is added to the disparity map.
And the 3D reconstruction (d) works quite well, despite the presence of some errors, as can be seen in Figure 2.

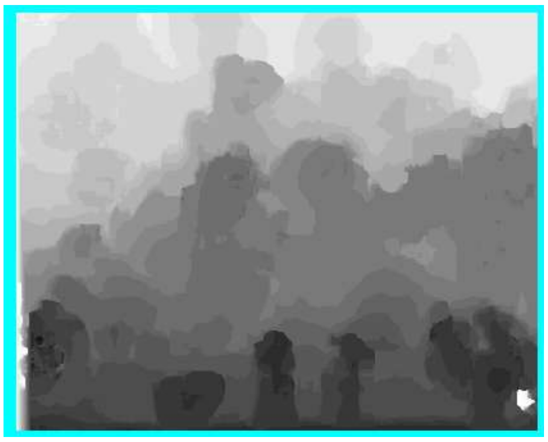
Using a larger window (9 instead of 4) leads to less obvious errors in the dense disparity map, as can be seen in Figure 3. This is simply because the NCC obtained by averaging over a larger window is more robust than with a small window.

Using a lower value of `nccSeed` (0.75 instead of 0.95) leads to the results in Figure 4. We notice more errors in the final disparity map, which makes sense, since we accepted as seeds pixels for which the best NCC is low.

We also used the program with another set of images, as can be seen in Figure 5, using pictures from Google Earth. The result is quite good there as well.


(a) Dense


(b) Seeds


(c) Propagation

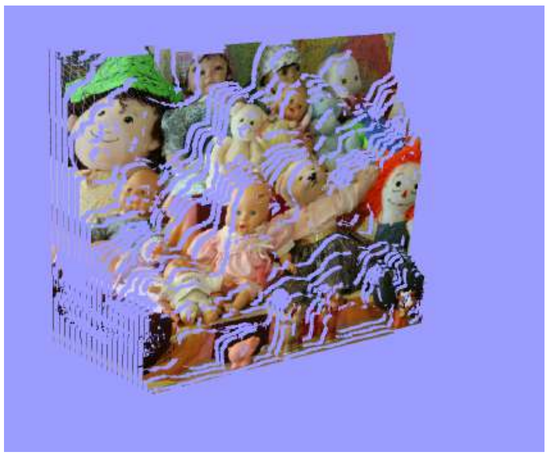
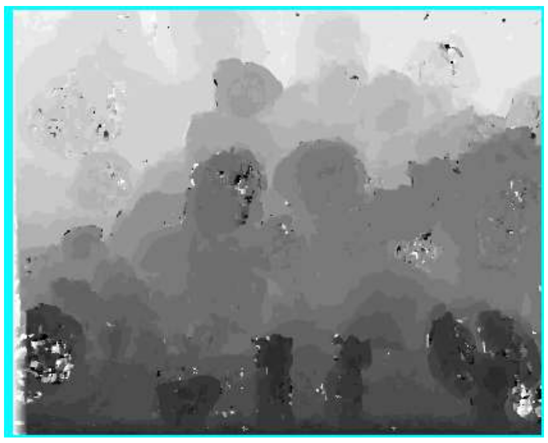
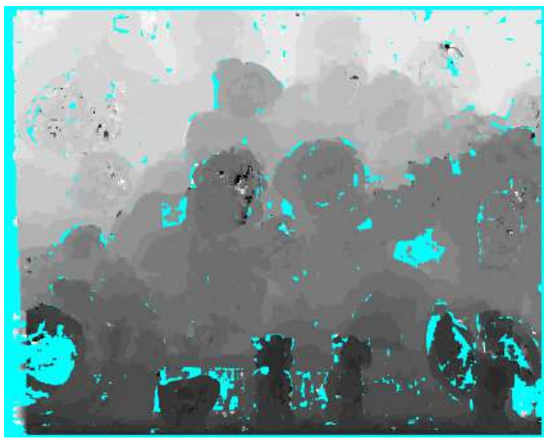
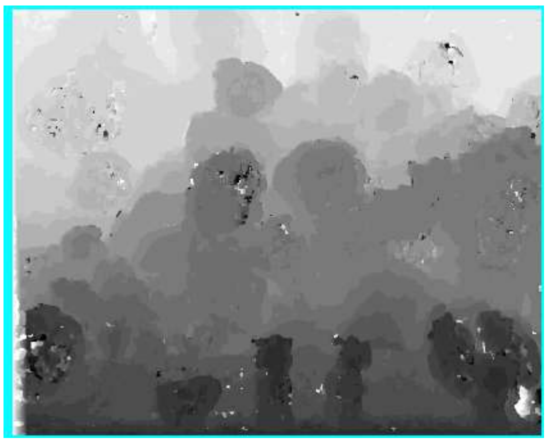

(d) 3D

Figure 3: Experiments on the given images, with a larger window


(a) Dense


(b) Seeds


(c) Propagation

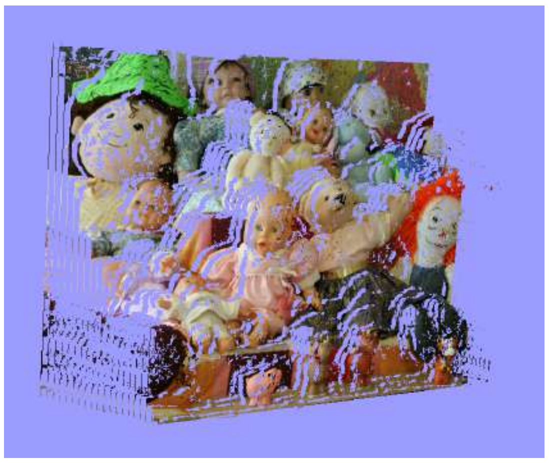
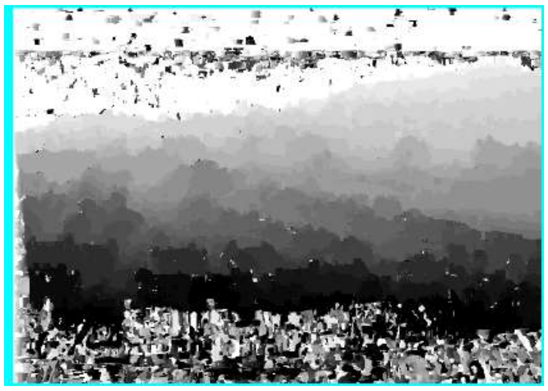
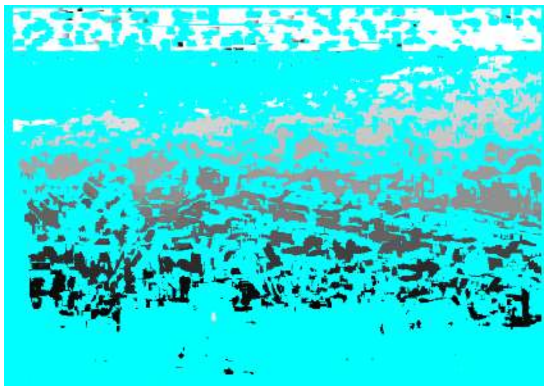

(d) 3D

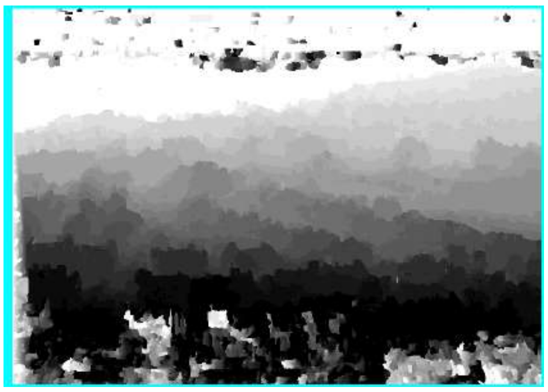
Figure 4: Experiments on the given images, with a lower value of `nccSeed`

(a) Image 1

(b) Image 2


(c) Dense


(d) Seeds


(e) Propagation

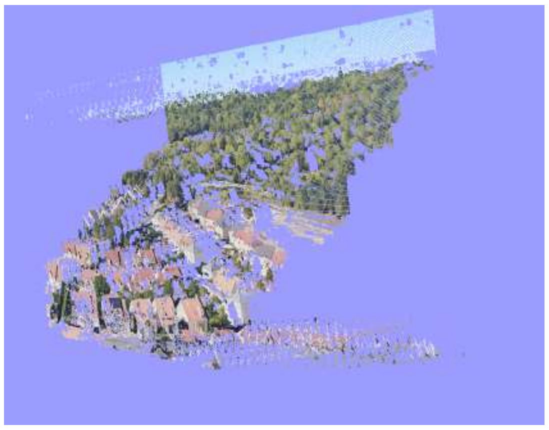
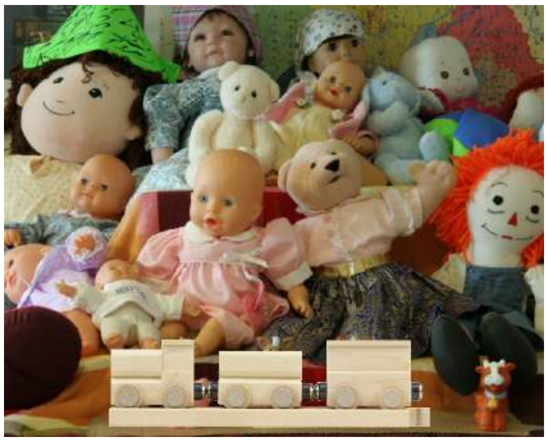
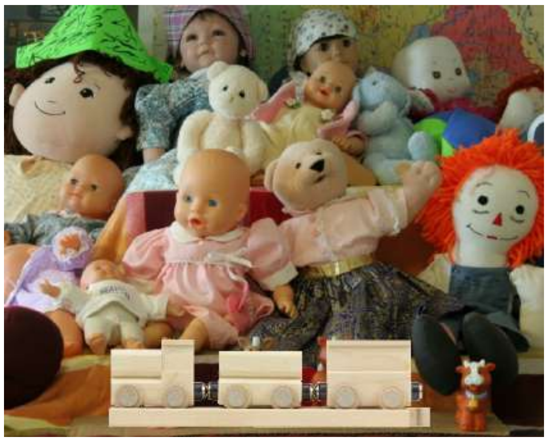

(f) 3D

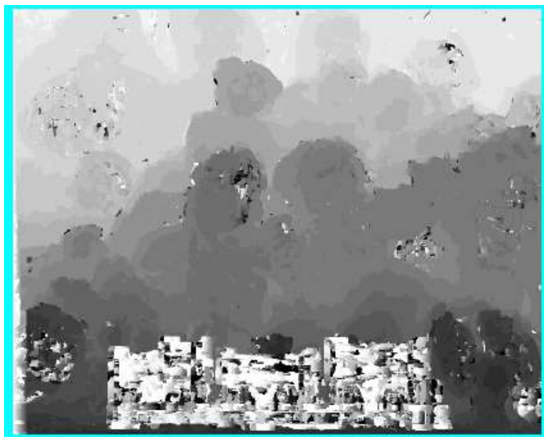
Figure 5: Experiments on other images

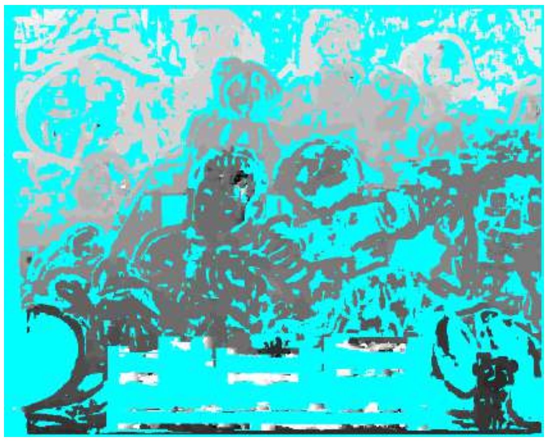
3.2 Limitations

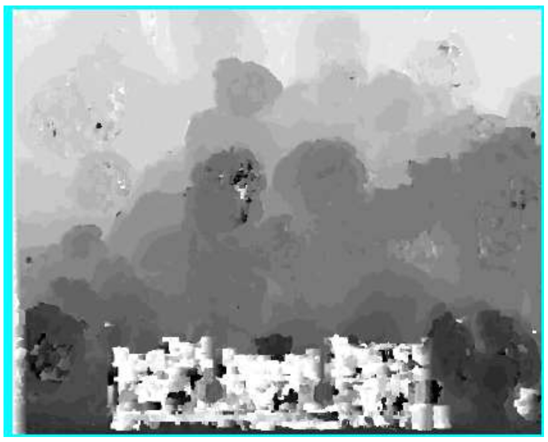
As mentioned in the lecture, if an object moves as the camera moves, then the result will not be good. This is illustrated in Figure 6, where we added a train in the image. This train moves at the same speed as the camera (that is, we put it exactly in the same position in terms of pixels in both images), and in the final estimation, it is considered to be far away from the camera. This is because it does not move between the two pictures, which would indeed be the case for an object far away.


(a) Image 1


(b) Image 2


(c) Dense


(d) Seeds


(e) Propagation

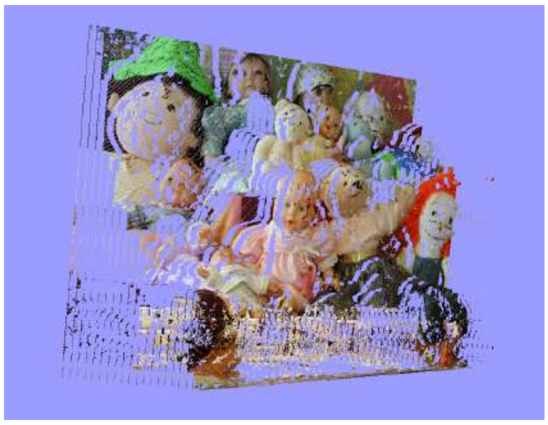

(f) 3D

Figure 6: A fail case

We also notice that not many points from the train were taken as seeds. This is explained by the fact that the train is mostly a uniform area on the picture, so the NCC will often be zero (since we have $I_L(p+r) \approx \overline{I}_L^{W(p)}$ for $r \in W$).
We had noticed that for some object on the left of the image, and we can also see it in Figure 5 in the sky.

On my laptop, the execution took approximately 2 minutes and 20 seconds (approximately 1 minute and 7 seconds for the "dense" part, 1 minute and 7 seconds as well for the "seeds" part, and a few seconds for propagation and 3D reconstruction), and even longer for a window of size 9 instead of 4.
So this takes some time, but I assume it can be optimized: we compute correlations using *for* loops, but this operation can be parallelized, so the total time can probably be reduced a lot.