REPORT HW1 – SIMONE BOSCOLO

INTRODUCTION

The homework aim is to develop the three main functions of the Patch Match Stereo algorithm, presented during the class, which are:

- SPATIAL PROPAGATION
- VIEW PROPAGATION
- DISPARITY PERTURBATION (in our case, as required by the assignment: we deal only with disparities and not planes, it means that we are dealing with planes those are all parallel to the camera).

ALGORITHM IMPLEMENTATION

I have implemented two versions of the same algorithm which differ only for the implementation of the disparity perturbation function. On a later section, you can see the results and the differences among the 2 versions.

I will briefly describe the implementation used to develop each function.

SPATIAL PROPAGATION

GOAL: look in the pixel neighborhood if there is a disparity that improves the matching cost.

PSEUDO-CODE

INPUT:

- Pixel **p=(x,y)**
- View **cpv**
- Iteration **iter**

```
1. IF (iter%2==0):
   THEN neighborhood pixels are the left and top pixel of p
   OTHERWISE, neighborhood pixels are the right and top pixel of p
2. FOR EACH pixel PIXEL p'
      a. IF p' is inside the image:
         THEN continue with the computation;
         OTHERWISE stop here.
      b. Retrieve disparity d and d' from p and p'
      c. IF m(p,d')<m(p,d) (m is the matching cost of a pixel and its disparity):
         THEN assign disparity d' to pixel p
```

VIEW PROPAGATION

GOAL: given a pixel in its view, look in the other view all the matching pixel if there is any disparity that improves the matching cost of the current pixel.

PSEUDO-CODE

INPUT:

- Pixel **p=(x,y)**
- View **cpv**

```
1. IF (cpv==LEFT VIEW):
   THEN other_view = RIGHT VIEW and sign = 1
   OTHERWISE other_view = LEFT VIEW and sign = -1

2. FOR EACH pixel p'=(x',y') with y'==y in other_view
      a. Retrieve disparity d and d' from p and p'
      b. IF (x'+sign*d'==x) [p' is a matching pixel for p]:
         THEN continue with the computation;
         OTHERWISE stop here.
      c. IF m(p,d')<m(p,d) (m is the matching cost of a pixel and its disparity):
         THEN assign disparity d' to pixel p
```

DISPARITY PERTURBATION

GOAL: given a pixel, randomly perturbate its disparity and if it improves the matching cost, update the disparity

VERSION 1

In this version we only check if the disparity perturbation provides a pixel that is inside the image, if not we reduce the perturbation.

PSEUDO-CODE VERSION 1

INPUT:

- Pixel **p=(x,y)**
- View **cpv**
- **max_perturbation** we can apply to the disparity
- **min_perturbation** we can apply to the disparity

```
1) IF (cpv==LEFT VIEW):
   THEN sign = -1;
   OTHERWISE sign = 11

2) WHILE max_perturbtion>min_perturbation
   a) Retrieve disparity d from p
   b) Set upper_bound=max_perturbation and lower_bound=min_perturbation
   c) Check if disparity perturbation doesn't exceed the interval [0, MAX DISPARITY]
         THAT IS:
      1) IF (d+upper_bound>MAX DISPARITY) THEN upper_bound=MAX DISPARITY-d
      2) IF (d-lower_bound<0) THEN lower_bound= - d
   d) Randomly sample a disparity perturbation from the interval
      [lower_bound, upper_bound], assign this value to delta and set
      new_disparity=d+delta
   e) IF (x+sign*new_disparity) is inside the image
      THEN continue the computation
      OTHERWISE max_perturbtion=max_perturbtion/2 and restart the cycle
   f) IF m(p,new_disparity)<m(p,d) (m is the matching cost of a pixel and its
      disparity):
      THEN assign disparity new_disparity to pixel p

   g) Set max_perturbtion=max_perturbtion/2
```

VERSION 2

In this version, if the disparity perturbation provides a pixel that is not inside the image, we try to fix the perturbation interval based on the position of the pixel.

In order to have a valid perturbation at each iteration of the while cycle, we need to have:

$$\begin{cases} \mathbb{I}: x + s * (d + \Delta) \geq 0 \\ \mathbb{II}: x + s * (d + \Delta) < col \\ s = \pm 1 \; (disparity\ sign, depends\ on\ the\ view) \\ 0 \leq x < col\ , integer\ (x\ coord.\ of\ the\ pixel) \\ 0 \leq d \leq MAX\ DISPARITY \\ -L \leq \Delta \leq U \; (L = lower\ bound; U = upper\ bound) \\ 0 \leq L \; \leq MAX\ PERTURBATION \leq MAX\ DISPARITY \\ 0 \leq M \; \leq MAX\ PERTURBATION \leq MAX\ DISPARITY \\ col = width\ in\ pixel\ of\ the\ image \end{cases}$$

We need to fix L and U, in order to have a valid Δ that respects equations $\mathbb{I}$ and $\mathbb{II}$.

Notice that we suppose that $x + d$ is always inside the image boundaries.

We have to consider 4 cases:

1) $View = RIGHT \Rightarrow s = +1$
   a) $\mathbb{I}: x + d + \Delta \geq 0\ is\ not\ valid \Rightarrow \Delta\ is\ too\ small \Rightarrow -L\ is\ too\ low\ as\ lower\ bound$
      $Therefore: x + d + \Delta \geq 0 \Rightarrow -x - d \leq \Delta \Rightarrow \boldsymbol{L = x + d}$
   b) $\mathbb{II}: x + d + \Delta < col\ is\ not\ valid \Rightarrow \Delta\ is\ too\ big \Rightarrow U\ is\ too\ high\ as\ upper\ bound$
      $Therefore: x + d + \Delta < col \Rightarrow \Delta \leq col - x - d \Rightarrow \boldsymbol{U = col - x - d}$
2) $View = LEFT \Rightarrow s = -1$
   a) $\mathbb{I}: x - d - \Delta \geq 0\ is\ not\ valid \Rightarrow \Delta\ is\ too\ big \Rightarrow U\ is\ too\ high\ as\ upper\ bound$
      $Therefore: x - d - \Delta \geq 0 \Rightarrow \Delta \leq x - d \Rightarrow \boldsymbol{U = x - d}$
   b) $\mathbb{II}: x - d - \Delta < col\ is\ not\ valid \Rightarrow \Delta\ is\ too\ small \Rightarrow L\ is\ too\ low\ as\ lower\ bound$
      $Therefore: x - d - \Delta < col \Rightarrow col - x + d \leq \Delta \Rightarrow \boldsymbol{L = col - x + d}$

If, after fixing upper and lower bound, the disparity perturbation still provides a pixel that is not inside the image, this means that the initial random disparity assignment assigns to the considered pixel a not correct disparity. Therefore, we try to assign randomly a correct value of the disparity, based on its position; that is:

1) $d = random\ value\ picked\ from\ interval\ [0, x]\ \ IF\ x + s * d < 0\ (only\ with\ pixels\ in\ the\ left\ view)$
2) $d = random\ value\ picked\ from\ interval\ [0, col - x]\ \ IF\ x + s * d \geq col\ (only\ with\ pixels\ in\ the\ right\ view)$

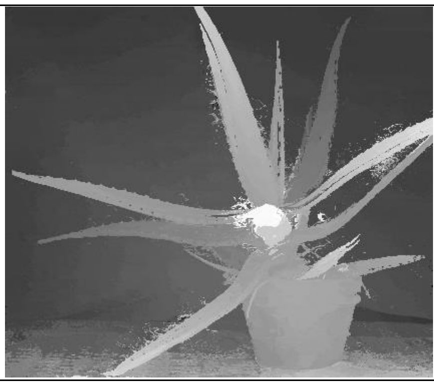
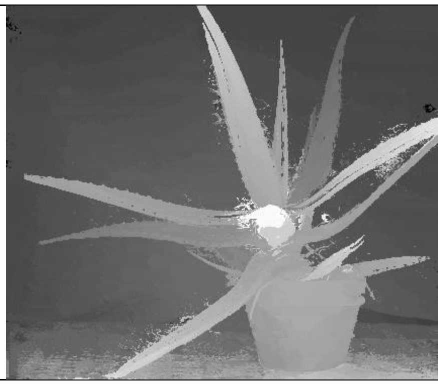
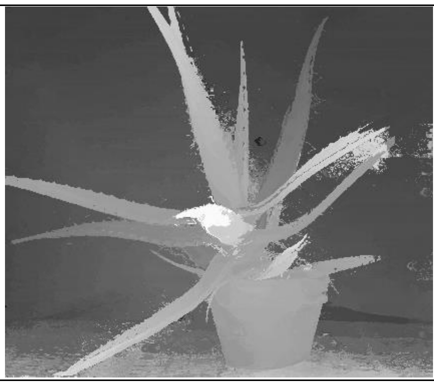
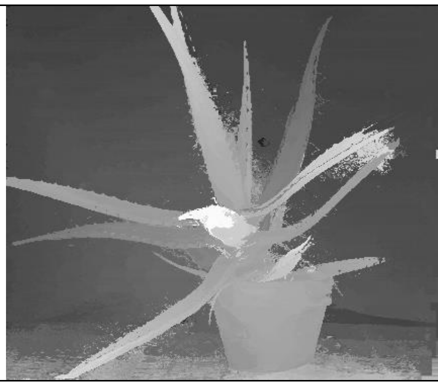
PSEUDO-CODE VERSION 2

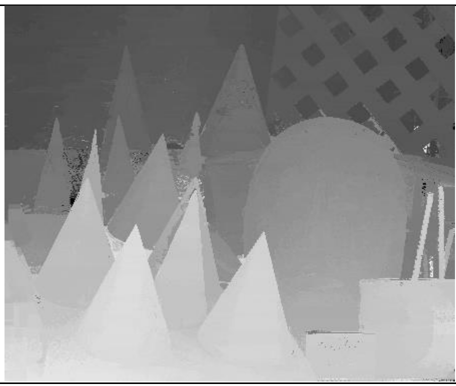
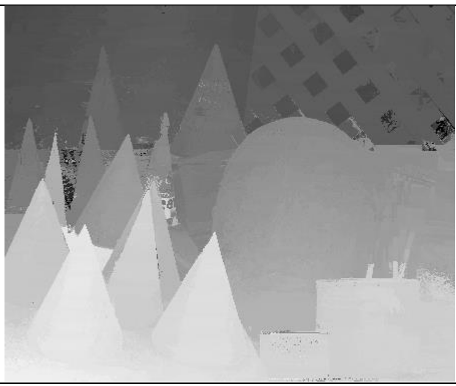
It is as the first version, but it modifies only the otherwise part on point e) in order to implement the consideration previously done

```
OTHERWISE:

1) IF (x+sign*new_disparity)<0 AND cpv==0 THEN  upper=x-disp
2) IF (x+sign*new_disparity)<0 AND cpv==1 THEN  lower=-(x+disp)
3) IF (x+sign*new_disparity)>=col AND cpv==0 THEN  lower=-(col-x+d)
4) IF (x+sign*new_disparity)>=col AND cpv==1 THEN  upper=(col-x-d)
5) Randomly sample a disparity perturbation from the interval
   [lower_bound, upper_bound], assign this value to delta and set
   new_disparity=d+delta
6) IF (x+sign*new_disparity) is not inside the image
   THEN:
   a) IF x+sign*d<0 THEN d=random in interval[0,x]
   b) IF x+sign*d>=col THEN d=random in interval[0,col-x]
   c) Update pixel p disparity with value d
   d) Restart the while cycle
   OTHERWISE: continue with point f) of the while cycle
```

RESULTS

ALOE		
LEFT DISPARITY MAP		
LEFT IMAGE MSE ERROR	19.0138	19.2034
RIGHT DISPARITY MAP		
RIGHT IMAGE MSE ERROR	26.0368	25.2436
	VERSION 1	VERSION 2

CONES		
LEFT DISPARITY MAP		
LEFT IMAGE MSE ERROR	34.2484	42.6166
RIGHT DISPARITY MAP		
RIGHT IMAGE MSE ERROR	37.8272	38.3596
	VERSION 1	VERSION 2

ROCKS1		
LEFT DISPARITY MAP		
LEFT IMAGE MSE ERROR	18.4193	26.3171
RIGHT DISPARITY MAP		
RIGHT IMAGE MSE ERROR	24.7972	24.9713
	VERSION 1	VERSION 2

FINAL CONSIDERATIONS

Unfortunately, the second version does not improve the final result, especially for the left view of the scene.

During the development of the assignment, I have tried to improve the results by modifying, with respect to the first version, the disparity perturbation function (the other two function respect the guidelines of the original Patch Match algorithm, so there are no reasons to modify them).

Therefore, I have decided to implement the random disparity perturbation in a way that the perturbation is always a valid perturbation and, in the case of pixel with disparity that doesn't match any pixel in the other view, I have decided to fix the pixel disparity to allow a match for each pixel. The disparity correction is especially done with pixels that are close to the edges of the images.

This modification doesn't improve the final results and this is correct because disparity defines the deviation of a pixel in the other view, but at the same time it defines the depth of a point in the scene (the Z coordinate). So, a point close to the edge does not have to be far from the camera (low disparity value), but it can be close to the it (high disparity value). For example, if we consider the pixel at position (0,0) on the left view, with the second version of my implementation it will never have a disparity different from zero; even if that pixel represents a point of an object that is close as maximum as possible to the camera.

I have decided to present either the results of the second version because I have discovered that it does not improve the final results (and the reasons why they are worse than the first version) while I was writing the report; therefore I wanted to show anyway the mathematical reasoning done during the development process of the code (and show earlier in this report) and to show, even if it is a fail, an attempt of improving the random disparity perturbation.

In conclusion, the first version is still the best one.