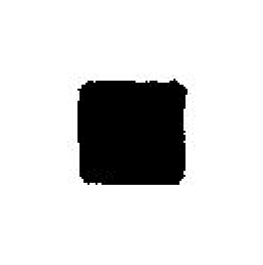
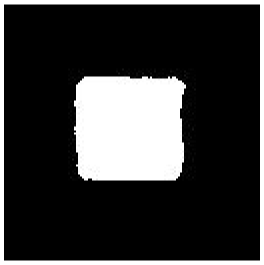
Problem Set 3: Window-Based Stereo Matching

Questions

1. Use the pair pair0-L.png and pair0-R.png - a central square moved 2 pixels horizontally:
   a. Implement the SSD match algorithm as function **`disparity_ssd`**`(L, R)` that returns a disparity image `D(y,x)` such that `L(y,x) = R(y,x+D(y,x))` when matching from left (L) to right (R).
   Function: `disparity_ssd`
   Output: Save disparity images:
   - D_L`(y,x)` [matching from left to right] as `ps3-1-a-1.png`:



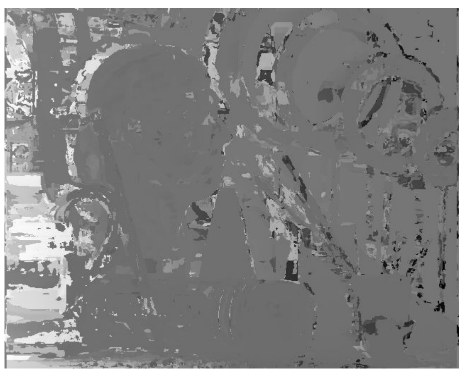
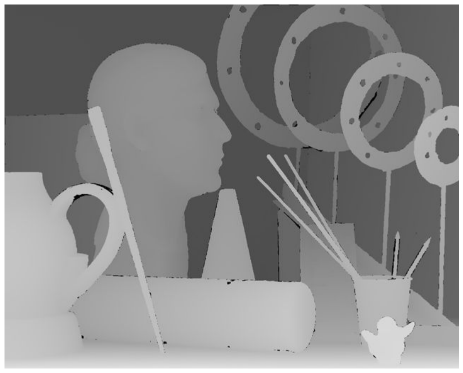
   Hard to see the background here, but it's the inverse of below
   - D_R`(y,x)` [matching from right to left] as `ps3-1-a-2.png`:



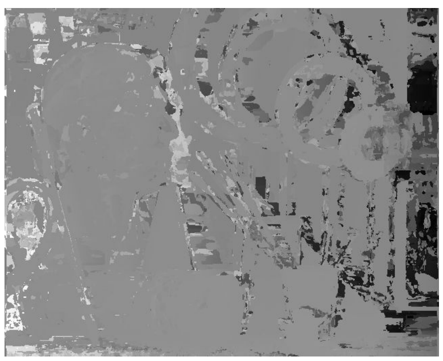
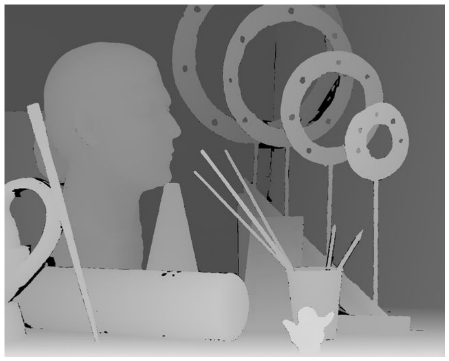
   These disparity images may need to be scaled and shifted to display/write correctly.

2. Now we're going to try this on a real image pair: pair1-L.png and pair1-R.png. Note that these are color images - so make sure you read in grayscale or convert.

   a. Again, apply your SSD match function, and create a disparity image `D(y,x)` such that `L(y,x) = R(y,x+D(y,x))` when matching from left to right. Also, match from right to left.
   Output: Save disparity images, scaling/shifting as necessary:
   - D_L`(y,x)` [matching from left to right] as `ps3-2-a-1.png`:

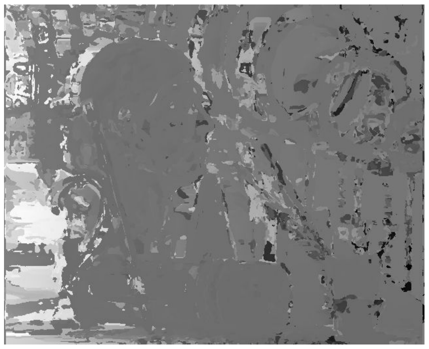



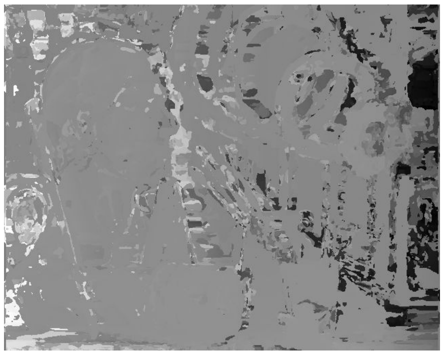
- D_R(y,x) [matching from right to left] as ps3-2-a-2.png:

b. In the input directory are ground truth disparity images pair1-D_L.png and pair1-D_R.png. Compare your results.
Output: Text response - description of the differences between your results and ground truth.
   i. I'll be the first to say that the results here are underwhelming compared to the ground truth images. There's a ton of noise in the image, and in general it misses a lot of the subtleties. However, that being said - it does successfully pull out some depth information. It's clear you can see the bust and the rings - looking closely we can just make out the loop, cylinder and cone. The paintbrushes looks like noise, and the supports of the hoops are pretty hard to see. It's interesting to note that the left image seems to have done a much better job on the left half of the scene and the reverse for the right image. My guess for this is based on the lighting of the shot.

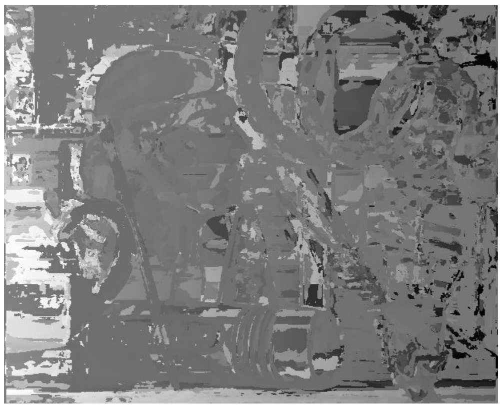
3. SSD is not very robust to certain perturbations. We're going to try to see the effect of perturbations:
   a. Using pair1, add some Gaussian noise, either to one image or both among pair1-L.png and pair1-R.png. Make the noise sigma big enough that you can tell some noise has been added. Run SSD match again.
   Output: Disparity images (D_L as ps3-3-a-1.png and D_R as ps3-3-a-2.png), text response - analysis of result compared to question 2.

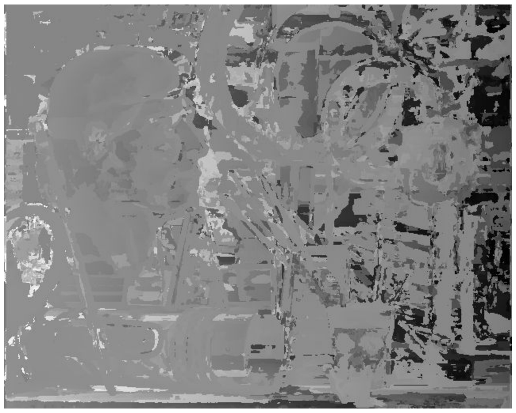

ps3-3-a-1.png


ps3-3-a-2.png

This was done by adding fairly significant noise to both images. (kernel of 5x5 sigma=11). However, the results are pretty similar. There is definitely a bit more noise in the second, looking at the edges of the bust for example, but because the first set was already pretty noisy, it's a tough thing to compare.

b. Instead of the Gaussian noise, increase the contrast (multiplication) of one of the images by just 10%. Run SSD match again.
Output: Disparity images (D_L as `ps3-3-b-1.png` and D_R as `ps3-3-b-2.png`), text response - analysis of result compared to question 2.


`ps3-3-b-1.png`


`ps3-3-b-2.png`

Again, it's tough to compare with the sheer amount of noise in the shots. These were created by adding contrast to the right image. Interestingly it seems to have helped a bit on finding the brushes in the scene, and adding more contour overall. It's also added a good bit of noise in the bust and a few other areas, but seems to have improved some of the processing on the right foreground.

4. Now you're going to *use* (not implement yourself unless you want) an improved method, a similarity measure called ***normalized correlation*** – this is discussed in the book. The basic idea is that we think of two image patches as ***vectors*** and compute the angle between them – much like normalized dot products.

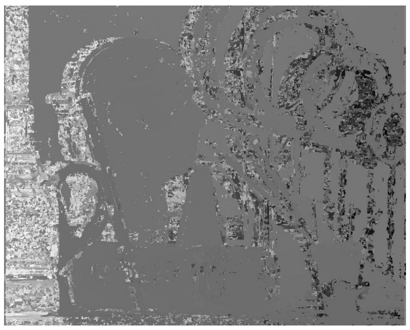
a. Using some form of normalized correlation, implement a window matching stereo algorithm. Again, write this as a function **`disparity_ncorr`**`(L, R)` that returns a disparity image `D(y,x)` such that `L(y,x) = R(y,x+D(y,x))` when matching from left (L) to right (R).
OpenCV has a variety of relevant functions and supported methods such as CV_TM_CCOEFF_NORMED, which implement correlation similar to:

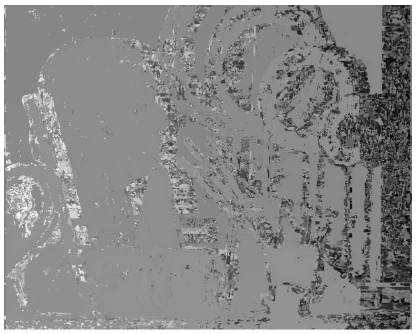
$$\gamma(u,v) = \frac{\sum_{x,y}\left[f(x,y)-\bar{f}_{u,v}\right]\left[t(x-u,y-v)-\bar{t}\right]}{\left\{\sum_{x,y}\left[f(x,y)-\bar{f}_{u,v}\right]^2\sum_{x,y}\left[t(x-u,y-v)-\bar{t}\right]^2\right\}^{0.5}}$$

You MAY use these built-in normalized correlation functions.

Test it on the original images both left to right and right to left (pair1-L.png and pair1-R.png).

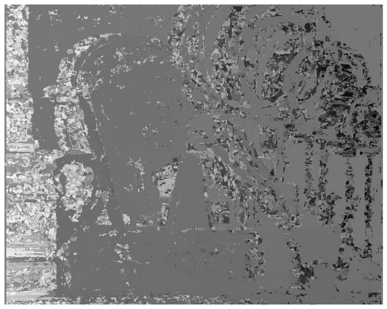
Output: Disparity images (D_L as `ps3-4-a-1.png` and D_R as `ps3-4-a-2.png`),
Text response - description of how it compares to the SSD version and to the ground truth.

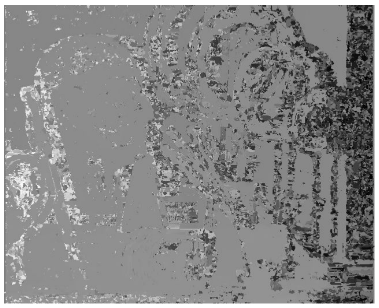

ps3-4-a-1.png

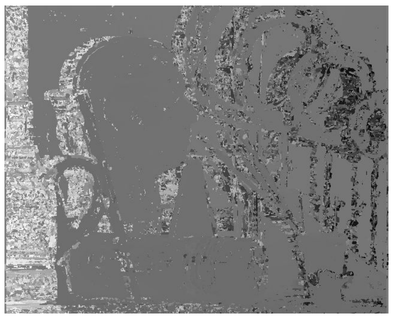

ps3-4-a-2.png

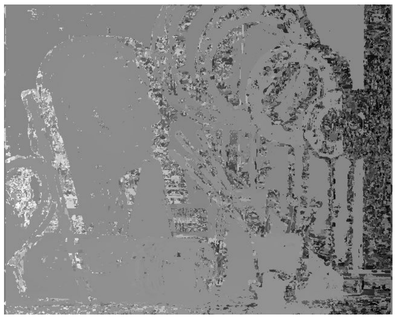
These were created with a window of 7 using normalized correlation. They are significantly cleaner than the previous SSD implementation, which is to be expected. The hoops are more defined and the bust is smoother. It picks up the foreground particularly well here. I played a good bit with the window size, and was challenged - getting higher in the numbers made for a softer, but more lossy looking image. Compared to ground truth… we're still pretty far off. There is still a lot of noise, and the brushes are tough to see. Overall, a step in the right direction though.

b. Now test it on both the noisy and contrast-boosted versions of pair1 from 3-a and 3-b.
Output: Disparity images (Gaussian noise: D_L as `ps3-4-b-1.png` and D_R as `ps3-4-b-2.png`; contrast-boosted: D_L as `ps3-4-b-3.png` and D_R as `ps3-4-b-4.png`),
Text response - analysis of results comparing original to noise and contrast-boosted images.


ps3-4-b-1.png


ps3-4-b-2.png


ps3-4-b-3.png

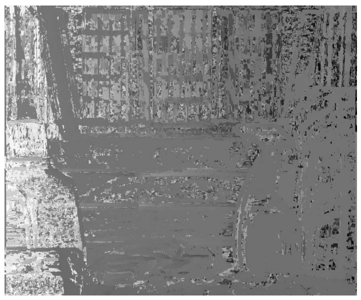

ps3-4-b-4.png

As you might have guessed, adding noise to the images (same as before 5x5, sigma=11) added a lot of noise to the output. The we lose a lot of definition in the bust and the background becomes very noisy. The contrast boost doesn't seem to have much of an impact in this situation. There's a slight change on the bust, but it's not particularly better or worse looking in my opinion

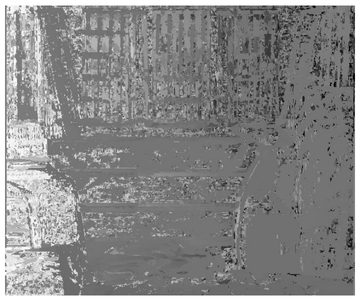
5. Finally, there is a second pair of images: pair2-L.png and pair2-R.png

a. Try your algorithms on pair2. Play with the images – smooth, sharpen, etc. Keep comparing your results to the ground truth (pair2-D_L.png and pair2-D_R.png).
Output: Disparity images (D_L as `ps3-5-a-1.png` and D_R as `ps3-5-a-2.png`), Text response - analysis of what it takes to make stereo work using a window based approach.

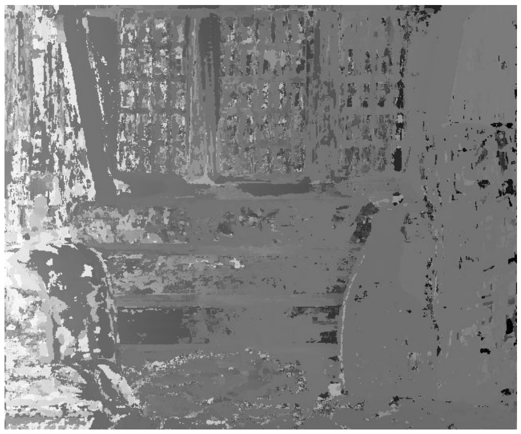
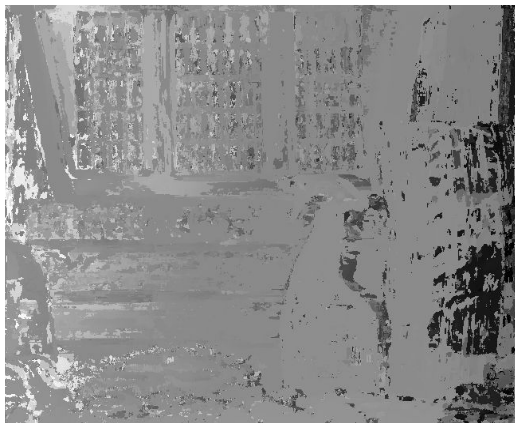
At first, I tried using ncorrelation to get the best image I could, but the end result wasn't great. I tried blurring and filtering before and after - nothing really seemed to get me closer to ground truth. I think the big thing that's needed is a way to threshold the responses - so things that are obviously noise are ignored. Given the tools we had on hand, I tried something a bit different here, to see if I could improve upon the standard approach. I realized that pair2 had a lot of distinct colors to it, so I thought that maybe I could leverage the color channels to get a result. Using each of the channels, I got slightly different results:


Blue channel

Green Channel

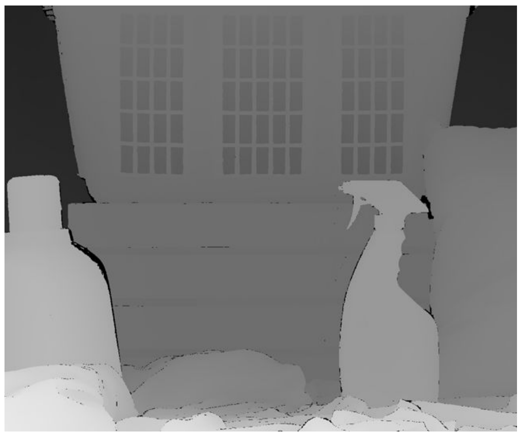
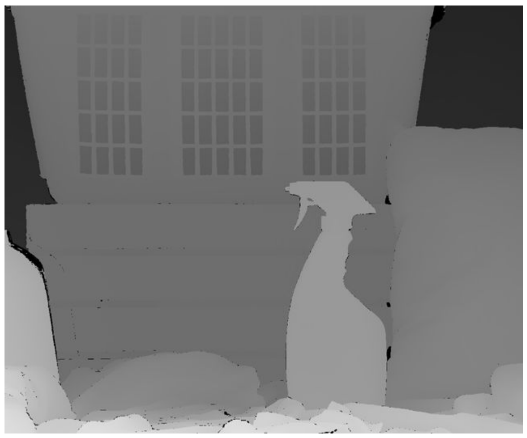

Red Channel

After that, I decided to try to modify SSD to use let each channel vote for the closest match, since I had a good 'scoring' vector. The channel that had the lowest ssd, got to fill in D. It led to an interesting result:

ps3-5-a-1

ps3-5-a-2

Obviously, we’re still pretty far from ground truth, but it’s interesting to note that the different color channels had different successes in determining similarity, and by combining this, I ended up with an image that did a very good job at finding the basket, and much of the foreground.