Part A:

Bilateral Filtering:

Bilateral Filtering takes a weighted average intensity of a pixel and the pixels around it, applying a mask to the original image. As a result, textures are softened/removed, and constant regions of colour/intensity appear. It is used in situations where a cartoonish effect or smoother image is desirable ([3]). The weighting may be based on the Gaussian distribution. It is non-linear, and preserves edges well. There are some flaws:

- Staircase Effect: edges of the image show obvious steps, making the edge look like a staircase (can be resolved using edge blending/linear interpolation)([3])
- Gradient Reversal: new edges may be introduced into the image as a result

The openCV bilateral filter uses 2 gaussian functions: one for intensity, and one for distance. The width of these functions are determined by σ_{color} and σ_{space} respectively. Through these parameters, one can change how much each attribute affects the filter, with an increasing σ resulting in a larger effect. Increasing σ_{color} increases the blur in the image by decreasing the range of intensities in a given area of the image, and increasing σ_{space} sharpens edges. σ_{space} is normally set to be the standard deviation of the distances from the centre of a window, where d is the diameter, and is given by the user (inferred from [1]). σ_{color} is also set to be around 5-10% of the total range of values. σ_{color} can be experimented with to obtain different results, whereas σ_{space} is dependent on d.
For test1.png and test2.png, the bilateralFilter() function included in openCV was used. In the documentation, it is stated that for σ_{color} and σ_{space}, values of 10 would yield very little change, whereas values of 150 would create larger differences. Also, if d is set to a non-zero positive integer, that integer will be used as d, rather than computing d from σ_{space}. Otherwise, σ_{space} would be used to determine the window size.

Test1.png:

For this test, the bilateralFilter() function included in openCV was used. Increasing σ_{color} makes the image look very smudged, and a lot of detail is lost. Increasing σ_{space} seems to have little to no effect on this image. Similarly, increasing d had very minimal effect; only slightly brighter darks and darker brights could be noticed. Increasing both σ_{color} and σ_{space} evenly results in the pixel intensity range being reduced, whilst retaining edge detail.
For a landscape image, keeping as much detail as possible is desirable. In this case, the image is also greyscale, so color is less of a concern, so long as we keep enough contrast between the brights and darks. At σ_{color} = 20 and σ_{space} = 70, I think these properties are achieved, such that the edges are slightly more defined, but not so harsh as to seem out of place. The effect is extremely subtle, but had σ_{color} been set higher, the image would seem clearly washed out.

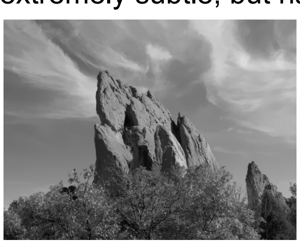

Fig.1 σ_c = 10 and σ_s = 10

Fig.2 σ_c = 10 and σ_s = 150

Fig.3 σ_c = 150 and σ_s = 10

Fig.4 σ_c = 150 and σ_s = 150

Test2.png:

Changing σ_{color} smoothes/smudges the image significantly, reducing the clarity of the image. Changing σ_{space} makes certain edges slightly more obvious. This is very noticeable in the hair of the model in the top left of the image. It also flattens the overall color of the image slightly - the model's cheeks become less pink. Increasing σ_{color} and σ_{space} evenly flattens the colours in the image whilst retaining sharpness.

For a portrait image, smoothing the skin whilst keeping color would be important, and I found that setting σ_{color} = 50 and σ_{space} = 30 gave desirable results - the hair becomes less detailed, but overall the skin looks much smoother. This is a case of some parts being over-blurred, due to the filter being applied uniformly across the entire image.

Fig.5 σ_c = 10 and σ_s = 10

Fig.6 σ_c = 10 and σ_s = 150

Fig.7 σ_c = 150 and σ_s = 10

Fig.8 σ_c = 150 and σ_s = 150

Part B:

Joint Bilateral Filter:

The Joint Bilateral Filter is similar to the regular Bilateral Filter in that it applies the same mask to an image, and requires the same gaussian functions to calculate said mask. The main difference is the use of two source images - a flash and a no-flash image, both taken in rapid succession. The RGB values of the flash image is then used in creating the mask, rather than the no-flash image. This mask is then applied to the no-flash image. The goal behind this is to improve the no-flash image, using information from the flash image, whilst retaining a similar colorspace; since the flash image has defined edges and minimal noise, that information can be transferred to the no-flash image such that edges are more defined and the image is less noisy, which is a common trait to images taken in the dark.

The joint bilateral filter is most useful in improving the clarity of a low-light image, whilst maintaining the same color profile it would have without the flash. The concept of the joint bilateral filter can be carried across into different use cases; a specific example includes red-eye correction in images - we can maintain the same detail from the flash image, but remove the red-eye generated by the flash image by transferring that detail to an image without flash/red-eye ([2]). Another example is in white balancing - using the flash and no-flash pair, we can infer ambient illumination in the image and hence balance the image most appropriately. Although unable to completely eliminate noise, the joint bilateral filter does reduce the impact of it to some extent. It is flawed, however, when the flash creates strong shadows that weren't present in the non-flash image, which may introduce new edges to the final image.

Test 3:

My implementation takes 4 parameters: a non-flash image, a flash image, a window radius r, and σ_{color}. I have tested 6 different values for r, and 5 values for σ_{color}. The window size is calculated as d = 2r+1. I used the Gaussian Function found in [4] (slide 18). Window size starts to show a clear effect at a size of 9 and above, with increasing window sizes showing more noise reduction. The r values I tested were 2, 4, 6 and 8, and the σ_{color} values are 1, 3, 5, 10 and 20. I've opted to exclude the results for 1 and 3 for most of the r values as they show little to no change.

Fig.9 r = 2, σ_{color} = 1

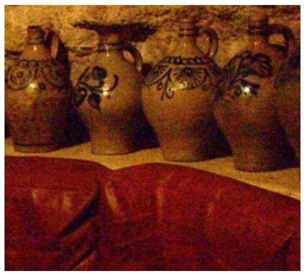

Fig.10 r = 2, σ_{color} = 20

Fig.11 r = 4, σ_{color} = 5

Fig.12 r = 4, σ_{color} = 10

Fig.13 r = 4, σ_{color} = 20

Fig.14 r = 6, σ_{color} = 5

Fig.15 r = 6, σ_{color} = 10

Fig.16 r = 6, σ_{color} = 20

Fig.17 r = 8, σ_{color} = 5

Fig.18 r = 8, σ_{color} = 10

Fig.19 r = 8, σ_{color} = 20

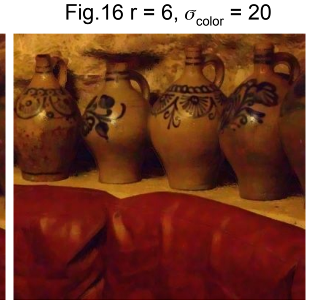

Fig.20 r = 11, σ_{color} = 5

Fig.21 r = 11, σ_{color} = 10

Fig.22 r = 11, σ_{color} = 20

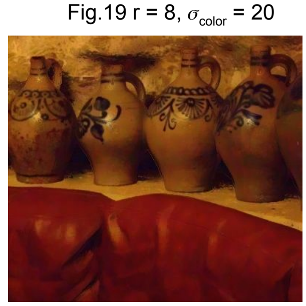

Fig.23 r = 15, σ_{color} = 5

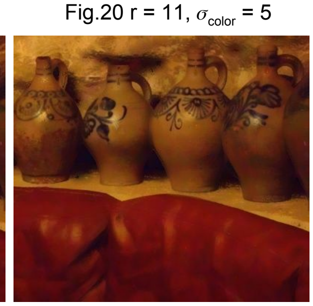

Fig.24 r = 15, σ_{color} = 20

As depicted above, the images where r is larger show increasingly less noise. This is because the window size was too small to clear any noise; the noise likely couldn't be contained in a 5x5

pixel window. However, past a certain point the image begins to seem clearly unrealistic, with parts of the image being too smooth, and some edges seeming to have a bright outline around them. Also, we begin to see that shadows in certain areas of the image become too defined, and start to look like their own features. This is especially visible in the handle of the far left vase, where a black spot grows into a black line as r is increased.
Going from σ_{color} = 10 to σ_{color} = 20 often shrunk the features in the images processed with a larger window. This is because increasing σ_{color} in turn increases the effect the surrounding pixel intensities have on the main pixels, hence it is likely to have a greater change in intensity/color. Of the values for r and σ_{color} I've tested, I believe an r value around 12 and a σ_{color} value around 12 would create the most desirable image, with a balance of noise reduction, edge clarity and realism.

Fine-tuned images:

Fig.25 σ_c = 20 and σ_s = 70

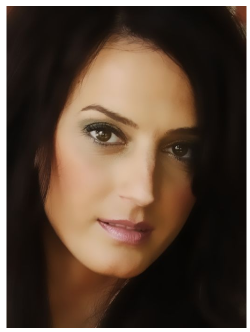

Fig.26 σ_c = 50 and σ_s = 30

Fig.27 r = 12, σ_{color} = 12

Reference List:

[1] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe and K. Toyama. "Digital photography with flash and no-flash image pairs." In ACM Trans. on Graphics, vol. 23, no. 3, pp. 664-672, 2004
[2] E. Eisemann and F. Durand. "Flash photography enhancement via intrinsic relighting." In ACM Trans. on Graphics, vol. 23, no. 3, pp. 673-678, 2004
[3] P. Kornprobst, "Limitations? - A Gentle Introduction to Bilateral Filtering and its Applications", Siggraph 2007, http://people.csail.mit.edu/sparis/bf_course/slides/09_limitations.pdf, 2007
[4] F. Durand, R. Raskar, S. Paris, S. Bae, "Bilateral Filters - Digital Visual Effects", In CSIE NTU Lecture Slides, https://www.csie.ntu.edu.tw/~cyy/courses/vfx/10spring/lectures/handouts/lec14_bilateral_4up.pdf, accessed Nov 2018

Appendix

Fig.1 σ_c = 10 and σ_s = 10

Fig.2 σ_c = 10 and σ_s = 150

Fig.3 σ_c = 150 and σ_s = 10

Fig.4 σ_c = 150 and σ_s = 150

Fig.5 σ_c = 10 and σ_s = 10

Fig.6 σ_c = 10 and σ_s = 150

Fig.7 σ_c = 150 and σ_s = 10

Fig.8 σ_c = 150 and σ_s = 150

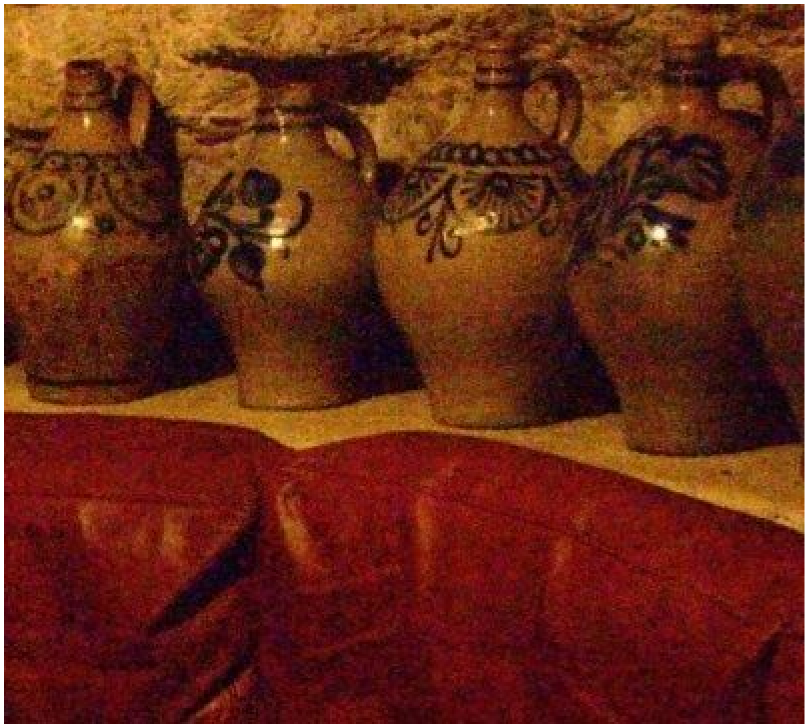

Fig.9 r = 2, σ_{color} = 1

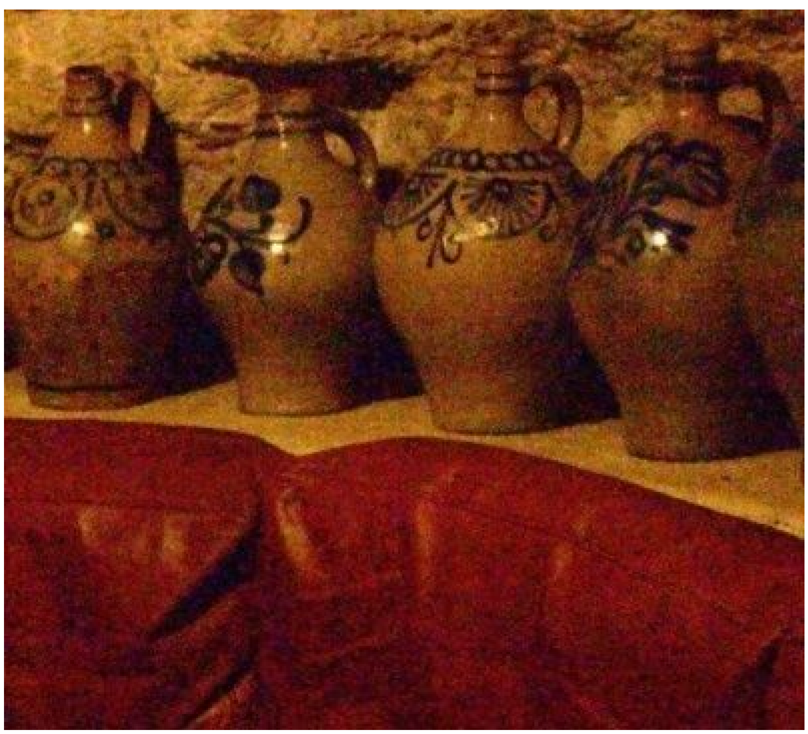

Fig.10 r = 2, σ_{color} = 20

Fig.11 r = 4, σ_{color} = 5

Fig.12 r = 4, σ_{color} = 10

Fig.13 r = 4, σ_{color} = 20

Fig.14 r = 6, σ_{color} = 5

Fig.15 r = 6, σ_{color} = 10

Fig.16 r = 6, σ_{color} = 20

Fig.17 r = 8, σ_{color} = 5

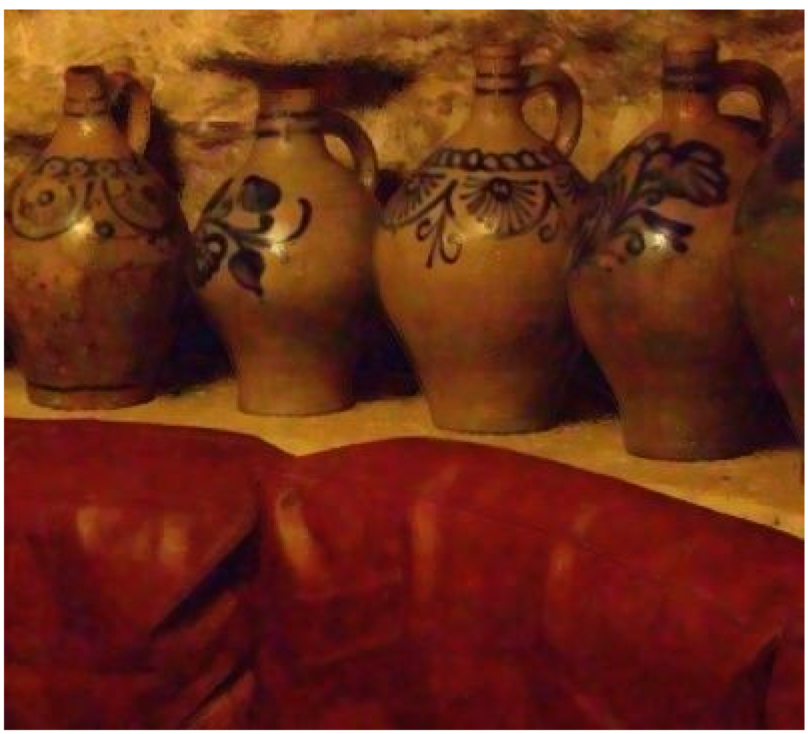

Fig.18 r = 8, σ_{color} = 10

Fig.19 r = 8, σ_{color} = 20

Fig.20 r = 11, σ_{color} = 5

Fig.21 r = 11, σ_{color} = 10

Fig.22 r = 11, σ_{color} = 20

Fig.23 r = 15, σ_{color} = 5

Fig.24 r = 15, σ_{color} = 20

Fig.25 σ_c = 20 and σ_s = 70

Fig.26 σ_c = 50 and σ_s = 30

Fig.27 r = 12, σ_{color} = 12