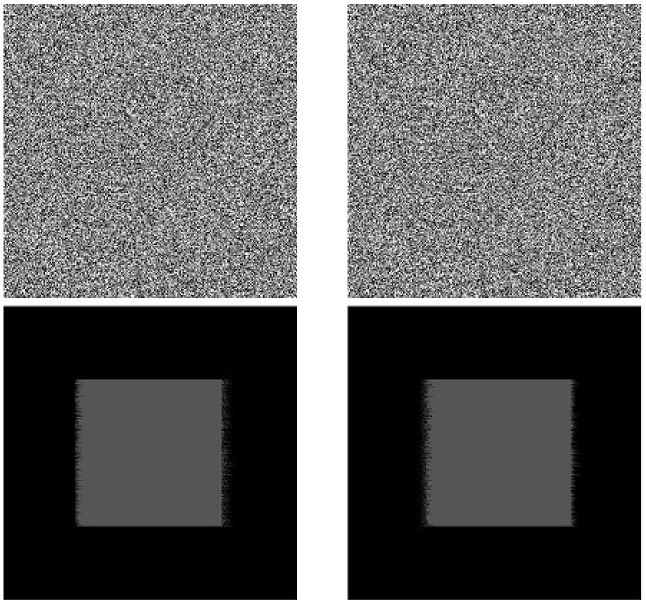
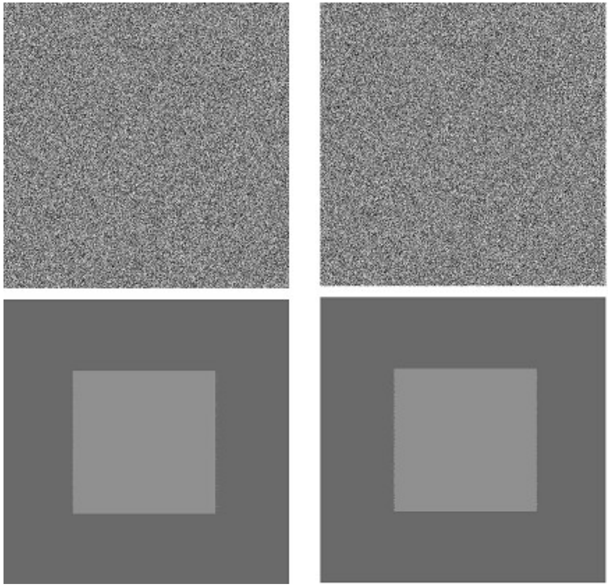
Depth Maps with Dynamic Programming

The algorithm matches pixels from both pictures from left to right. The disparity algorithm has matching errors for the binary random dot Stereo gram due to the pixels having either 0 or full intensity. The algorithm might accidentally match a pixel that does not correspond between the two images as it checks two pixels at a time according to their intensities and as these are either 0 or 255(full) the probability that a pixel is matched when it shouldn’t be is high. Accidental matching in turn skews the disparity data which results in slight blurring of the lines between different images. This is illustrated by the example below. First the algorithm is applied to a binary random dot Stereo gram and then a non-binary random dot Stereo gram.



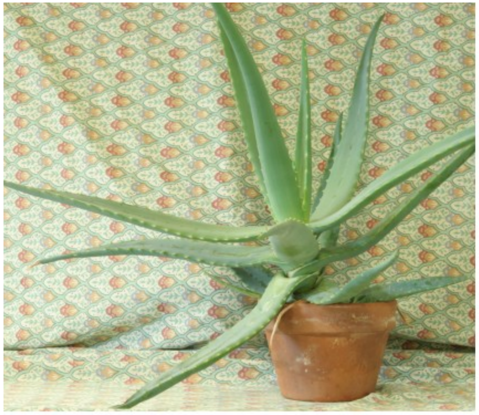
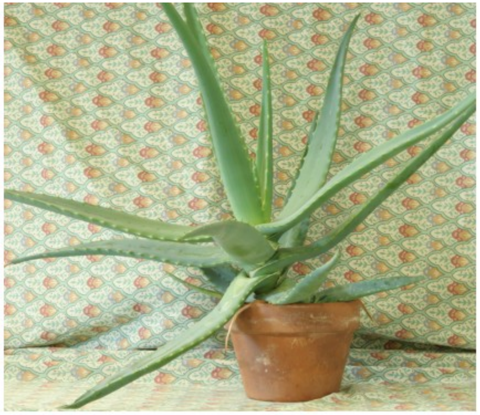
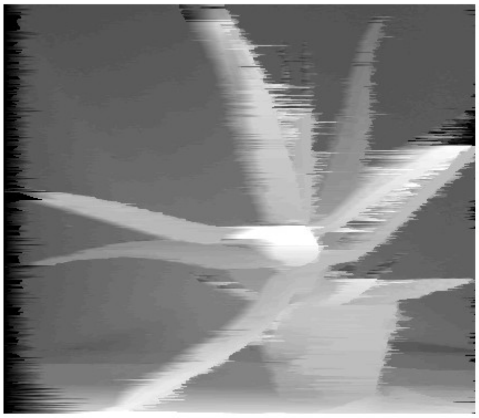
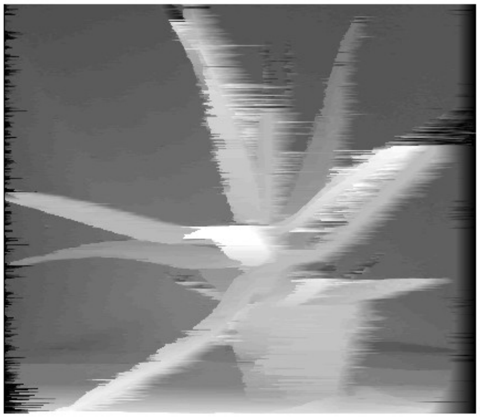
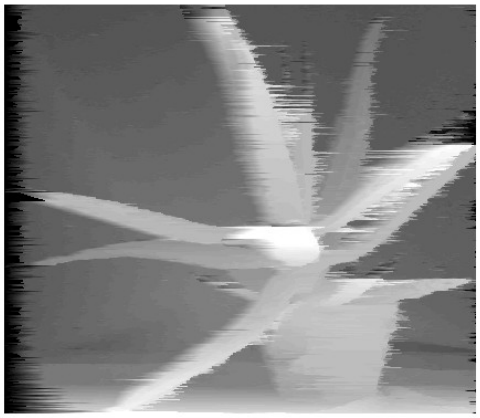
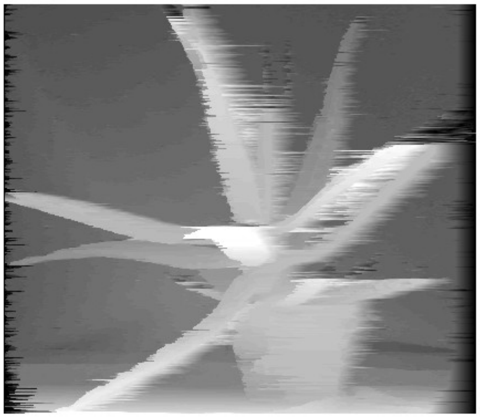
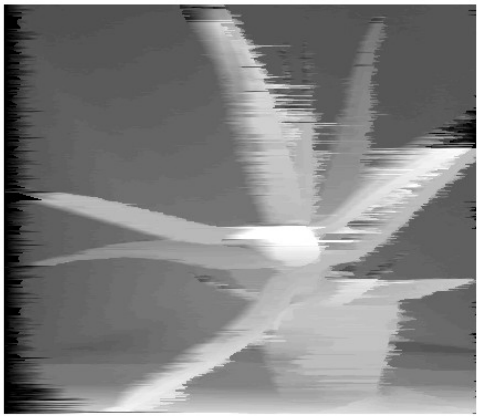
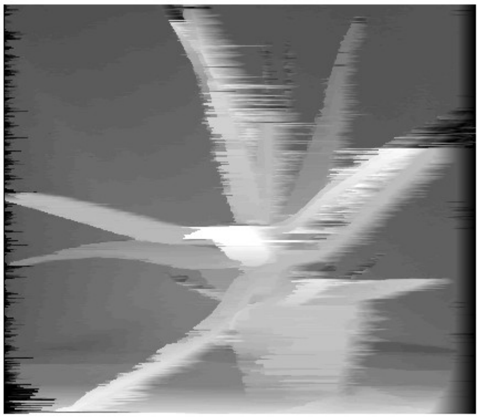
Depth Maps with Dynamic Programming



As visible the binary RDS has matching errors around the box inside the image while the non-binary RDS does not.

The Occlusion cost was varied from the original according to the equation which is 2.67 to 3.8 and 6. The resulting disparity maps are given below. Changing the Occlusion cost to 3.8 decreased some matching errors but changing it to 6 increased them well over what existed with the Occlusion 2.67. Occlusion is used to determine whether the difference is pixel values in two points is within acceptable variation which helps the backward pass to determine whether a pixel should be matched to the one is question or not. Varying the Occlusion cost would essentially vary the ‘acceptable range’ of difference between given pixels and in turn change whether a pixel should be matched or not. For a graph of Occlusion cost against Matching errors we’d see a curve with a lowest point. This lowest point would be the optimal setting.

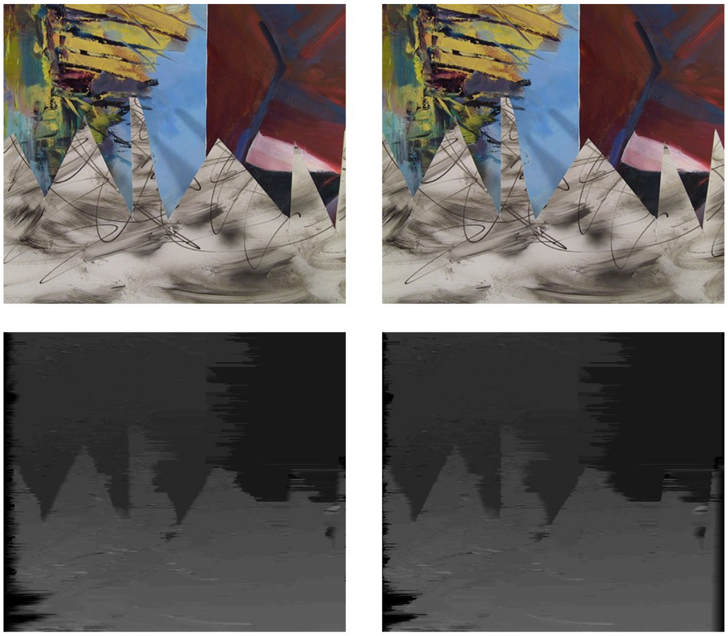
Depth Maps with Dynamic Programming

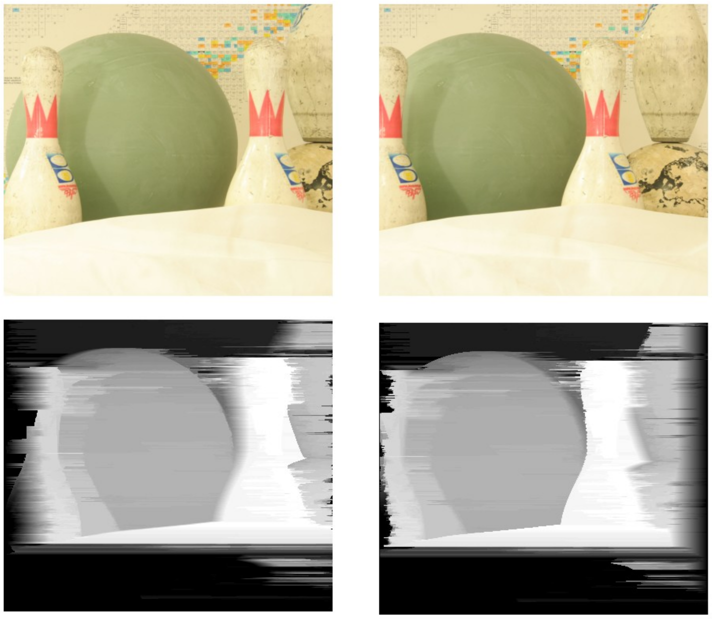









Depth Maps with Dynamic Programming

The equivalent to Occlusion in String matching is the cost of insertion, substitution or deletion of a character in the String. The Cost Map is made of String matching would depend on these costs and would then be used to deduce the fastest way of matching a String.

The following images are other results of using the algorithm on stereo pairs.





Depth Maps with Dynamic Programming

Due to performing the algorithm on pixels after getting an equivalent value in greyscale there are matching errors working on surfaces with similar color, if a whole pixel line has approximately the same color the algorithm would not to be able to recognize it as a surface at all. Resulting in matching errors. The algorithm also has errors when the final plane of an image is assessed going from right to left, leaving the rest of the image black in color.

Running the algorithm in one color causes a few errors but does work to differentiate objects if they are in the same y co-ordinate of the image.