# 3D Reconstruction of Satellite Images

## Table of Content

# 1. Introduction to the topic

Our project is the 3D reconstruction of satellite images. When satellites take pictures of the earth, only a projection of the 3D world on a 2D image plane takes place. This comes along with loss of information. However, those are usually important aspects for further analyzation of the captured area. To recover the lost information, researchers established multiple approaches.

The motivation for this project is trying to implement the reconstruction described in this paper http://dev.ipol.im/~facciolo/pub/CVPRW2017.pdf by ourselves. For that reason, we use the image dataset of the IARPA Multi-View Stereo 3D Mapping Challenge that took place in 2016. For verifying our implementation, we intend to use the s2p python pipeline as reference.

# 2. Used Methods

The steps we took for our implementation are explained in the following.

## 2.1. Preprocessing: Adjustment of image data types

Images obtained by satellites are often stored in special data types for geo data. One of them is the geoTiff format which is an extension to the regular tiff format. They have usually a large resolution since they contain a lot of information. GeoTiff images do not only store the regular image data but also information on the satellites position and its orientation. This is necessary for further usage of the images, since only with this information the images can be mapped together at the right position.

Our dataset had the additional data, in our case rpc values, stored in a separate text file. For further usage it was necessary to add the rpc values to the geoTiff image.

## 2.2. Preprocessing: Warping of images to real world coordinates

Satellites take pictures with different orientations, e. g. the direction north for an image taken from satellite A may be the direction west for an image taken from satellite B. Therefore it is necessary to warp the images to real world coordinates. This ensures the images to have the same orientation which is inevitable for the next step.

## 2.3. Adjust image sizes

Due to the warping, it might occur that height and width of image A and image B are not equal. In this case, the smaller image is padded until the sizes are the same.

## 2.4. Block Matching Algorithm

By using a block matching algorithm, blocks of image A and B are compared. There are two hyperparameters: the number of disparities and the block size. The first parameter limits the number of disparities. Landscape can have a lot of disparities and therefore the default value has proven to be best suited. The second parameter scales the size of blocks that are to be compared. A too small block size results in a noisy disparity map. Otherwise, a too big block gives blurred results. Also here, the default value is best suited.

## 2.5. Disparity map

The disparity shows the differences between image A and image B.

## 2.6. Visualization

With the information of the disparity map, a surface plot or a point cloud is created. Both show the height differences. This can be either be done by processing a single image pair or alternatively by processing multiple image pairs. This results then in a better approximation of the real world since more cameras from different positions contain more information.

# 3. Used Material

## 3.1. Programming

We did the programming in python 3.9. All used libraries can be found in the source code.

## 3.2. Verification Method

For verifying our method, we used the s2p pipeline. Here, python version 3.8 is necessary. The library can be found at https://github.com/centreborelli/s2p.

## 3.3. Image Dataset

For performance testing we used the image dataset of IARPA Multi-View Stereo 3D Mapping Challenge. It can be downloaded by using the command `aws s3 ls --no-sign-request s3://spacenet-dataset/`.

# 4. Results

The following chapter shows first the results of our own implementation and then a comparison with the s2p pipeline.

## 4.1. Own implementation

Our implementation works well on the tested dataset. *Figure 1: Point Cloud of one stereo pair, Figure 2: Surface Plot of one stereo pair* and *Figure 3: Surface Plot of multiple stereo pairs* show this.

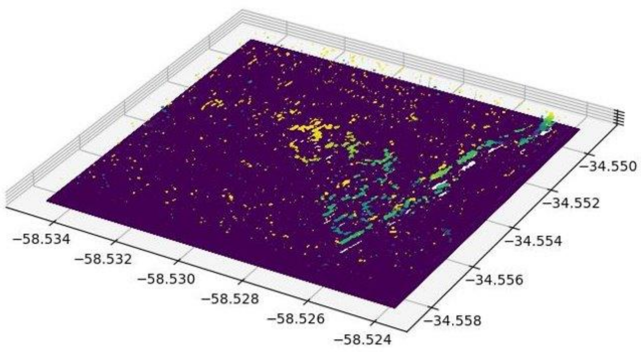


*Figure 1: Point Cloud of one stereo pair*

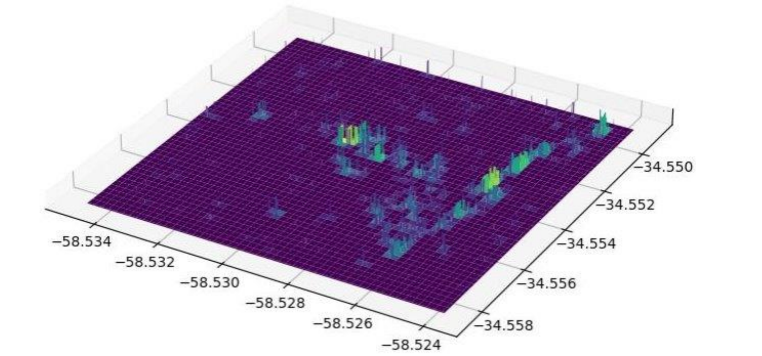


*Figure 2: Surface Plot of one stereo pair*

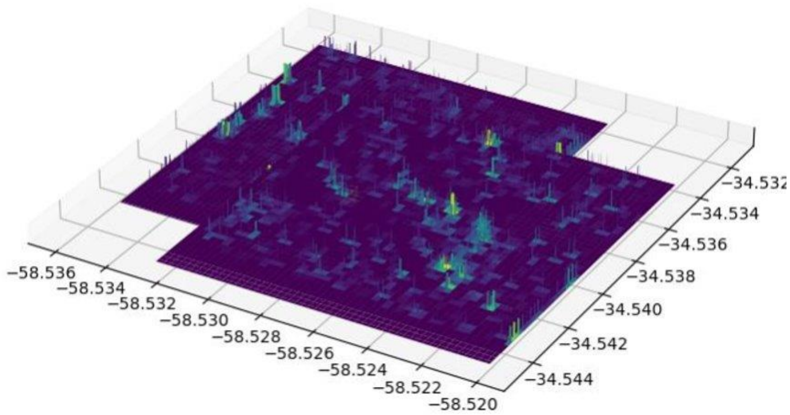


*Figure 3: Surface Plot of multiple stereo pairs*

## 4.2. Comparison with s2p

The following shows a comparison of our method with the usage of the s2p pipeline.

The pipeline can only be used with a small selection of images from the mentioned dataset. This is due to various input restrictions of the pipeline. As the documentation is very limited, it is not mentioned there. For most images, an error message saying the “range of interest differs or is completely masked” appears. Since the documentation does not include any value ranges, we tried out plenty of image combinations until no error message appeared anymore.

The used input images are

"Challenge1_LowRiseBuildings_CROPPED_03APR15WV031000015APR03140238-P1BS-500497283030_01_P001_________AAE_0AAAAABPABR0.tif"

and

"Challenge1_LowRiseBuildings_CROPPED_23OCT15WV031100015OCT23141928-P1BS-500497285030_01_P001_________AAE_0AAAAABPABO0.tif".

*Figure 4: Disparity Map of 2 Input Images* shows the two mentioned input images and the corresponding disparity map.

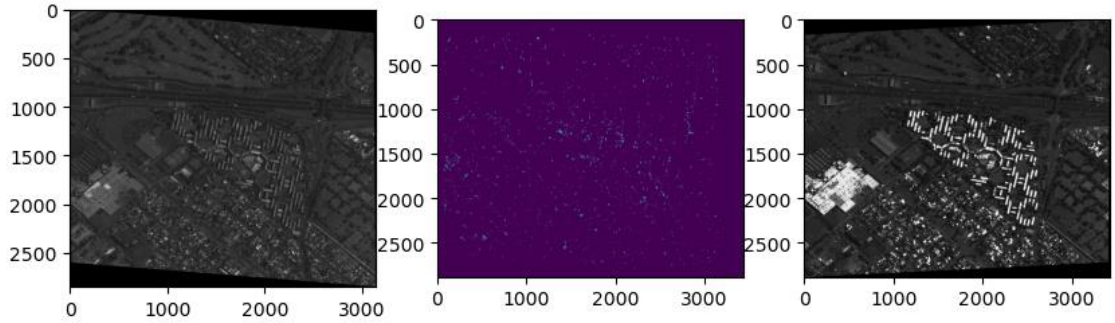


*Figure 4: Disparity Map of 2 Input Images*

In *Figure 5: Comparison of Surface Plots* the results of the s2p pipeline and our own implementation are displayed. The surface plots show no similarity except for the one yellow elevation spot in the middle of the image. By comparing them with the original images above, we cannot explain where the green rectangles in the s2p result come from.

We decided not to establish any state-of-the-art metric for method verification due to two reasons: since the s2p pipeline does not work for most combination of images we only have a sample of one which makes it impossible to make a clear statement. Second, the surface plots of the s2p pipeline and our own implementation are not similar at all. Hence it is not possible to make a comparison between the two methods due to lack of examples.

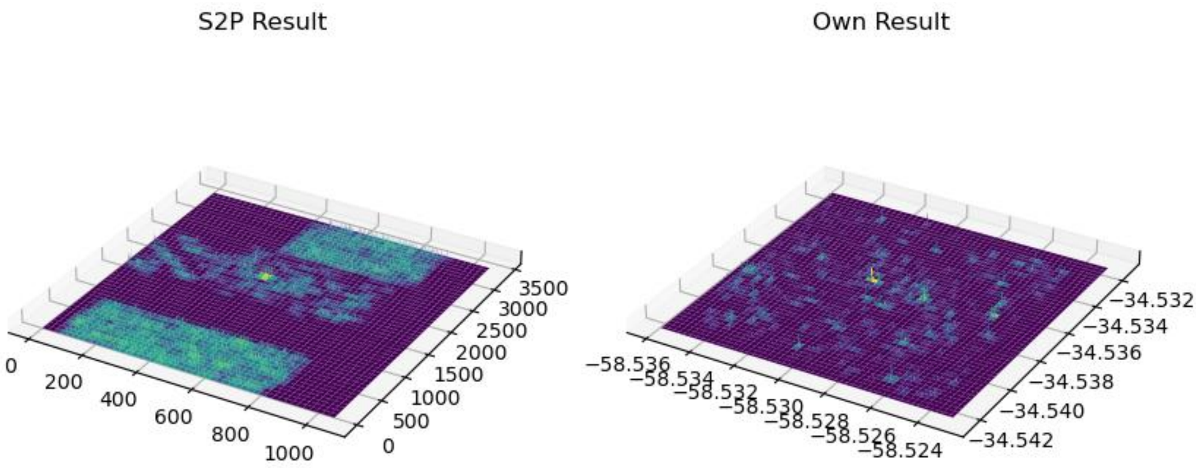


*Figure 5: Comparison of Surface Plots*

# 5. Responsibilities

The initial idea was that each of us would test a different approach for the 3D reconstruction.

We did not expect that satellite images are stored in specific data formats that differ so much from regular images and according to this, did not know that we would need to spend a lot of extra time in understanding them.

Since the data was not stored in a format that we could easily use for further processing we first had to make the data fit our purposes. We all tried different ways for combining the data but for one alone it was impossible to find a solution. In the end, the most efficient way was implementing all together at one script at the same time. This way, we could all focus on the same approach and find the mistakes, that one alone easily could have overseen.

Since it took us a long time to find out how to use satellite data, it was not possible to split the work in the beginning. After completing the preprocessing task, we did not have enough time to implement three approaches, thus we focused together on finding a way that works out fine.

We really did mostly everything together in online meetings, but while Katja was focusing more on doing the research, Stefan was the main programmer and Benjamin was the pair-programmer.

# Table of Figures: