M4 Project
Week 4: Structure from motion

Group members:

José Manuel López, Alex Martin, Marcos V. Conde, Oriol Catalan Yarza

Group ID: 4

1. Triangulation

- **Create the function triangulate(x1, x2, P1, P2, imsize) that performs a triangulation with the homogeneous algebraic method (DLT)**

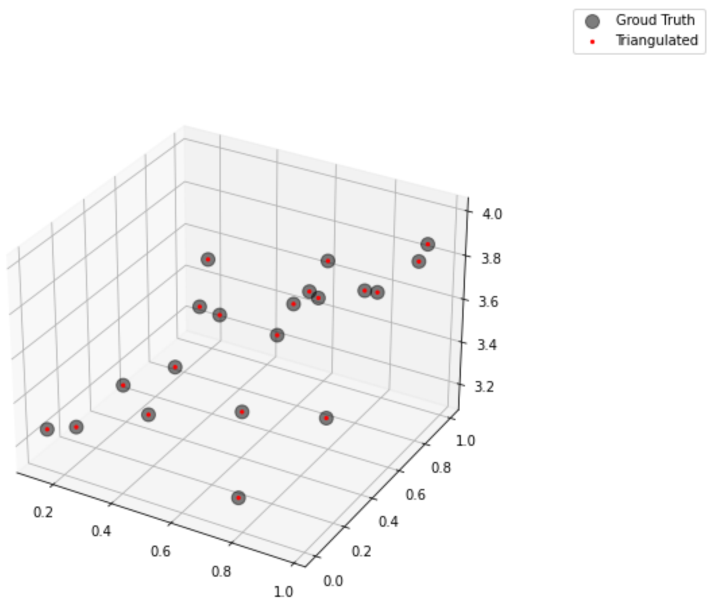
To perform the triangulation, we may use the points in images with different view in P3, and the camera matrices of the two cameras used.

With this information, we can construct a matrix A which once has been solved through SVD by extracting the last column of V we can obtain the points in the P4 which converted to euclidean coordinates are the points in 3D.

When we are using SVD on the Matrix A what we are doing is solving the following system of equations:

$$\begin{aligned} x(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{1\top}\mathbf{X}) &= 0 \\ y(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{2\top}\mathbf{X}) &= 0 \\ x(\mathbf{p}^{2\top}\mathbf{X}) - y(\mathbf{p}^{1\top}\mathbf{X}) &= 0 \end{aligned}$$

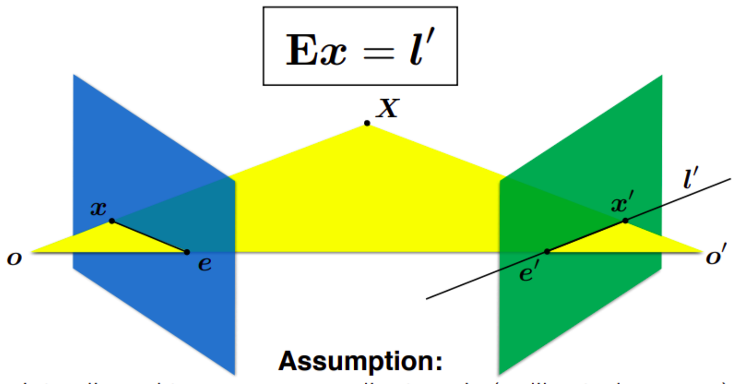
Which comes from the fact that the cross product between the point x1 and P1X has to be 0. Using also this condition applied to x2 we can obtain the same equation but with P2 and x2 and therefore build A. What we are using is the fact the we know how the points in 3D project into the image with the camera matrices and that we know the points of to points of view (x1,x2) to extract our incognita X.



Result of performing the triangulation with the random points

2. Reconstruction from two views

For this section, our goal is to perform an estimation of the 3D reconstruction based of two views in a practical case where the correspondences of the images contain outliers.



points aligned to camera coordinate axis (calibrated camera)

To achieve our goal, we need to complete some steps:

1. Find a set of point correspondences between the two images. These correspondences contain noise/outliers, therefore we cannot use Epipolar Geometry to calculate the Essential Matrix, but we can find the Fundamental Matrix.
2. Estimate the Fundamental Matrix relating the two camera Images. Since correspondences are noisy we will use the 8-point Algorithm in combination with the RANSAC algorithm, to find a good estimate of F.
3. Calculate the Essential Matrix from the Fundamental Matrix
4. Find the two Camera Matrices.

2.1 Estimation of the Essential matrix

To estimate the essential matrix from the fundamental matrix we just have to use the definition of both to know which operation has to be done.

$$F = K^{-T} \cdot E \cdot K^{-1}$$

So then just by isolating the E we have:

$$E = K^{T} \cdot F \cdot K$$

2.2 Estimation of the Camera matrices from the Essential Matrix

- **Estimate the camera projection matrices. Assuming the first Camera has a cannonical matrix you will obtain four possible Camera Matrices for the second camera, two rotations and two translations**
- **Please explain how do you obtain these estimates, not what algorithm or methodology you used to obtain them. i.e., What system of equations are you solving?**

To obtain the estimation of the camera matrices, first we have to assume that the first camera matrix is the canonical one. The other camera matrix is computed as described in the Zisserman book, chapter 9.6.2. The reason behind we can do that is the properties of the matrix E gives us more constrain, which we can use to extract the translation and rotation. The fact that E can be decomposed into a skew symmetric S and an orthogonal W matrix is the one used to extract the rotation and translation and therefore the camera matrices.

- **Why do you obtain 4 solutions? Why not only one?**

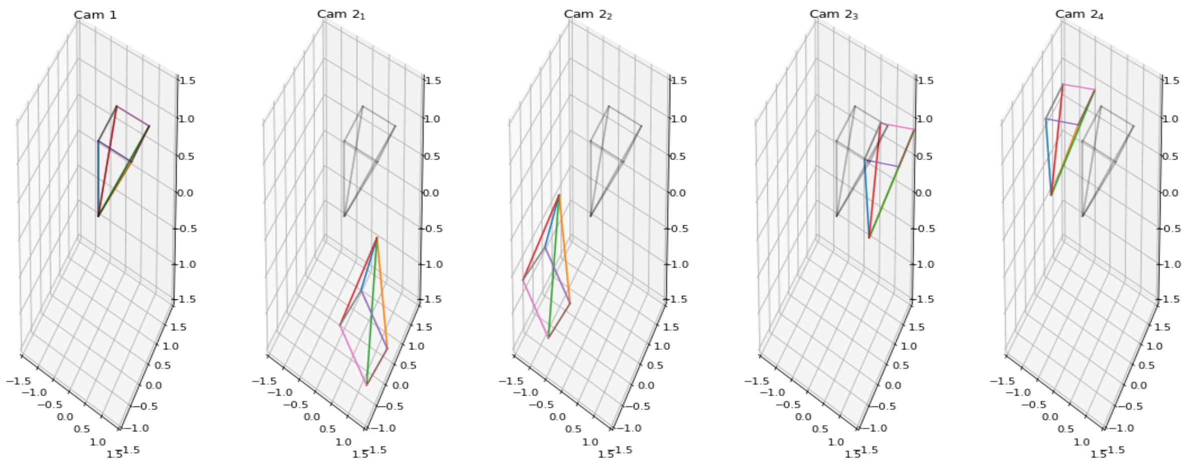
Because E has to have the values of the diagonal matrix of the SVD equal to one and the other one has to be 0 the Essential matrix has two different decomposition that comply with the constraints of E. Moreover, as the sign of E can not be determined when extracting the translation from the U matrix this can have 2 values plus and minus t. This gives you the 4 different possibilities.

- **Why is there a reprojection error ? From where comes the uncertainty ?**

The reprojection error comes from the fact that there can be distortions in the keypoints used to be compared with the projection of the 3D points to the image's back.

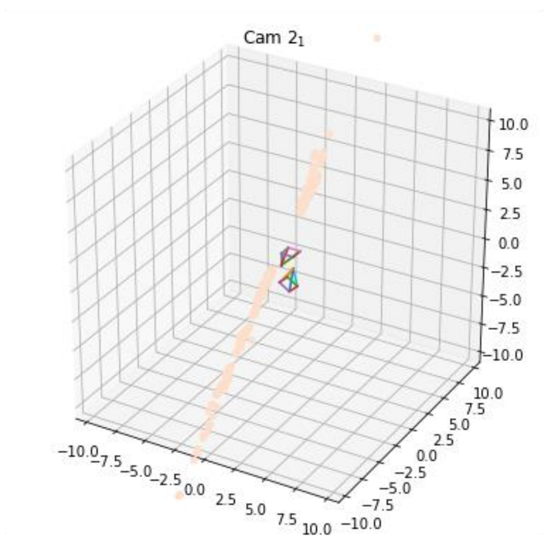
- Estimate the camera projection matrices. Assuming the first Camera has a canonical matrix you will obtain four possible Camera Matrices for the second camera, two rotations and two translations.

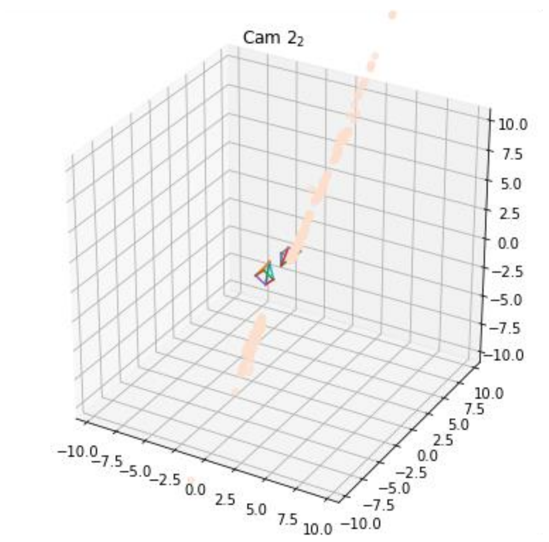
By using the essential matrix we obtained the four different camera matrices and this are the plots.

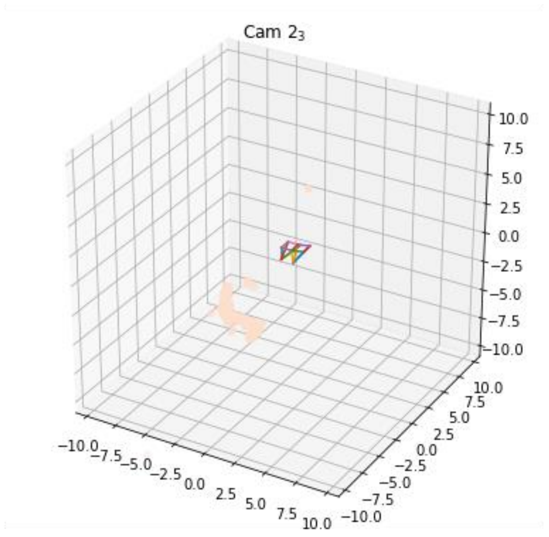


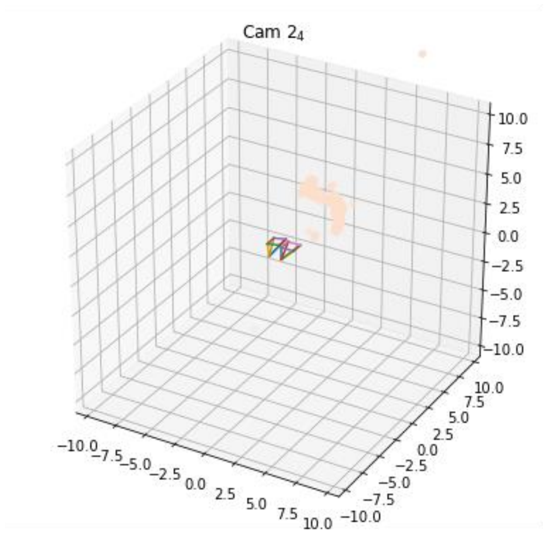
- Triangulate the correspondences and select the "optimal" camera, how do you select it ?

We used the previous triangulation function to estimate the 3D location of the keypoints from which we obtained this plots.





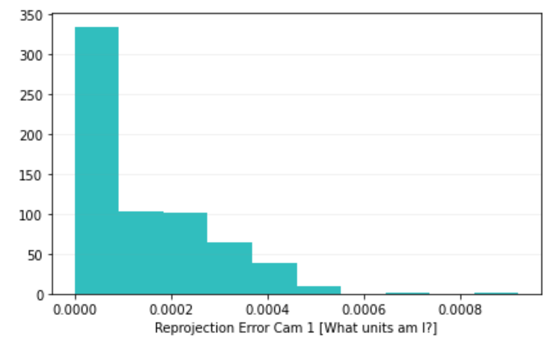


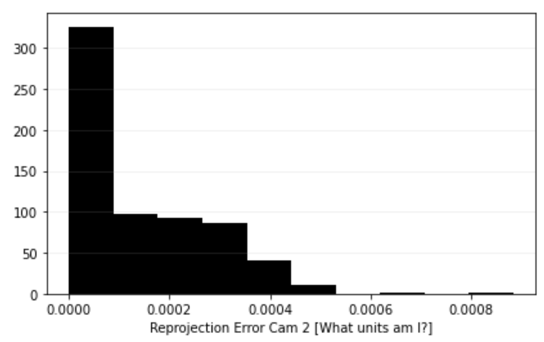


We didn't automatize the process of selecting which camera was the correct one but by looking at the plot we can observe that the Cam 24 plot is the correct one as the points appear to be in front of the 2 cameras.

- Compute the reprojection errors - Plot the histogram of reprojection errors, - Plot the mean reprojection error

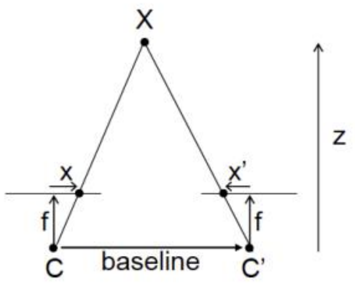
```
Mean reprojective error for cam1 = 0.02967202539504447
Mean reprojective error for cam2 = 0.03059344645817963
```





3. Depth map computation - Stereo Matching

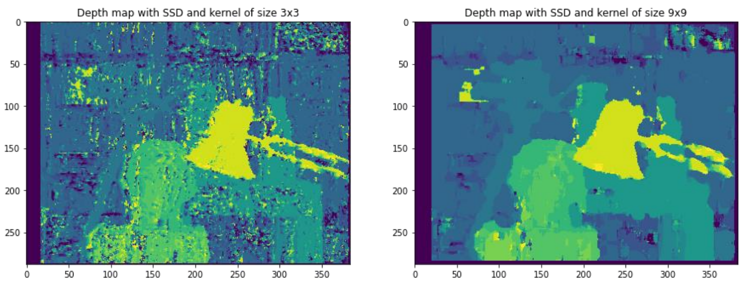
We implemented the stereo matching methods that used the SSD and NCC, but we couldn't implement the adaptative weights methods. The methods were implemented as explained in class estimating the correct disparity by using the match functions. The disparity that produced the lowest value was selected as the one for that point in the estimated image after using the selected window to compute the value of the matching function at that point. The values for the disparity are directly related to this geometric figure, which is what permits us to estimate the depth from two images.

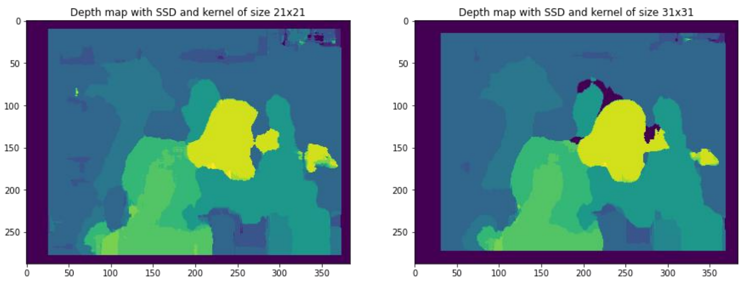


$$disparity = x - x' = \frac{baseline * f}{z}$$

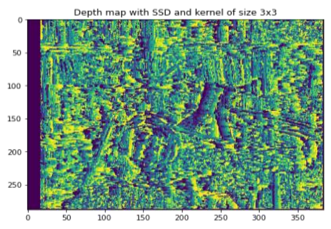
- Evaluate the results changing the window size (e.g. 3x3, 9x9, 21x21,31x31) and the Mean Square Error (MSE). Comment the results.

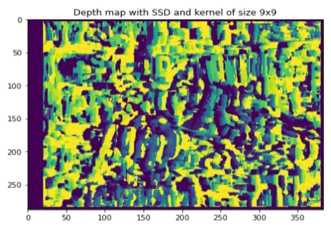
For the SDD

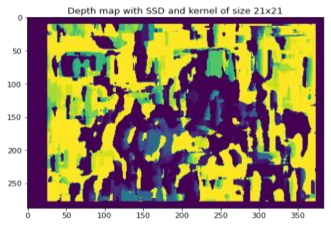


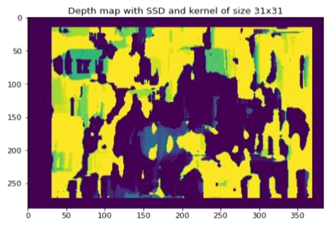


For the NCC:





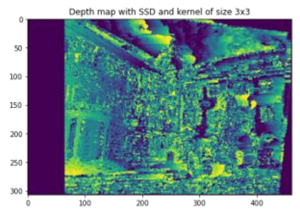


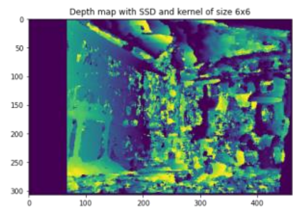


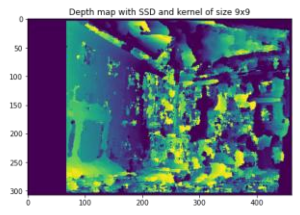
In both match function, we can observe the same pattern. When using smaller windows, we can observe more details, but also a lot of noise is present. In the case of the bigger windows, we can observe that the edges are lost, but the noise is avoided. When using the NCC we encountered not satisfying results, and we are not aware of what could be wrong as we tried different implementations, but we always obtained this result which we think it may not be the correct one.

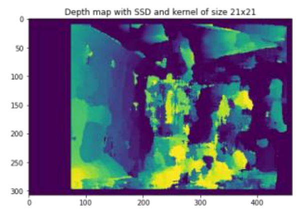
- Test the functions implemented in the previous section with the facade images. Try different matching costs and window sizes and comment the results.

With the SSD matching function, we tried a disparity of (0,32) and (0,64). Here we show the results for the (0,64) as they seem to be more interesting:

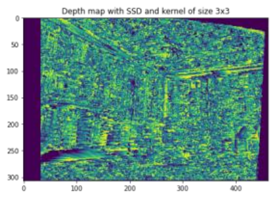


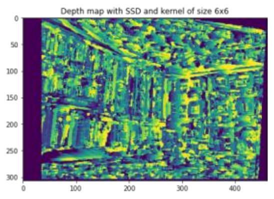


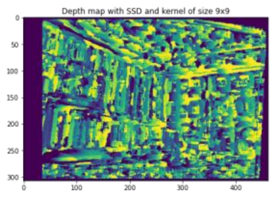


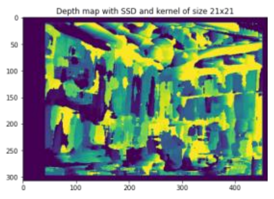


For the NCC we used disparity from 0 to 32 and the results are the following:









When using the stereo matching on the facade with SSD we obtain that the left facade in the map depth present the expected values as the more near points are brighter, but then it fades to 0 before arriving to the other part of the facade and then the middle of the map depth appears to have a lot of distortion. In the case of the 3x3 window size, we can observe a depth map that corresponds more to the expected from the photo, but the presence of noise is too much for estimating anything.

OPTIONAL - Depth Map Fusion Analysis

A Volumetric Method For Building Complex Models From Range Images

The paper presents a volumetric method to integrate range images to reconstruct some objects. The representation needs to accomplish some properties:

- Asymmetric Error Distributions from range images should be clearly reflected.
- Use all the available data, even redundant, to reduce noise.
- Incremental and order independent updating to select best scan orientation and get rid of bias.
- Time and space efficiency for practical usage in real world scenarios.
- Robust to outliers and systematic range distortions.
- No restrictions on topological types (Not assuming any genus).
- Ability to fill holes in reconstruction with plausible and aesthetically pleasant surfaces.

The presented algorithm employs a continuous implicit function D(x) that corresponds to the weighted signed distance of each point x to the nearest range surface along the line of sight to the sensor. A cumulative signed distance function D(x) and a cumulative weight W(x) are obtained for each voxel. Those functions are represented in a discrete voxel grid and then the isosurface corresponding to D(x) = 0 is extracted.

$$D(\mathbf{x}) = \frac{\Sigma w_i(\mathbf{x}) d_i(\mathbf{x})}{\Sigma w_i(\mathbf{x})} \qquad D_{i+1}(\mathbf{x}) = \frac{W_i(\mathbf{x}) D_i(\mathbf{x}) + w_{i+1}(\mathbf{x}) d_{i+1}(\mathbf{x})}{W_i(\mathbf{x}) + w_{i+1}(\mathbf{x})}$$

$$W(\mathbf{x}) = \Sigma w_i(\mathbf{x}) \qquad W_{i+1}(\mathbf{x}) = W_i(\mathbf{x}) + w_{i+1}(\mathbf{x})$$

di(x) and wi(x) are the signed distance and weight for the i-th range image. Di(x) and Wi(x) are the cumulative signed distance and weight functions after integrating the i-th range image.

Summarize the pipeline for 3D scenario:

- Set all voxel weight to 0 so new data will overwrite the initial grid values.
- Tessellate each range image with triangles from nearest neighbors on the sampled lattice (tessellating over step discontinuities must be avoided discarding triangles with edges that exceed a certain threshold).
- Update voxel grid after a range image has been converted to a weighted triangle mesh.
- Signed distance contribution is computed casting a ray from the sensor to each voxel near the range surface and then intersecting it with the triangle mesh).
- Weight is computed linearly interpolating the weights stored at the intersection triangle's vertices.
- With signed distance and weight, we can apply the update formula (cumulative).
- Zero crossing isosurface from the volumetric grid can be extracted at any point of the merging process (range images).

This algorithm reconstructs observed portions of the surface, but unseen portions will be observed as holes. One option for filling holes is to operate on the reconstructed mesh: if regions with holes are nearly planar this works well. Otherwise, volume will be used instead of the reconstructed mesh.

All points in the volume need to be classified as unseen, empty or near the surface. Holes in the surface are indicated by frontiers between unseen and empty regions. Surfaces placed at frontiers offer a plausible way to plug these holes. So, our algorithm extends:

- Initialize voxel space as unseen.
- Update voxels near the surface using continuous signed distance and weight values.
- Follow the lines of sight back from the observed surface and mark the corresponding voxels as empty.
- Perform isosurface extraction at the zero-crossing of the signed distance function. Additionally extract a surface between regions seen to be empty and regions that remain unseen.

Representation of the unseen and empty states using the function and weight stored on the voxel lattice. Unseen is D(x) = Dmax, W(x) = 0 and the empty state is D(x) = Dmin, W(x) = 0. This allows the usage of the usage of the same isosurface extraction algorithm without the restriction on interpolating voxels of weight 0. This extraction finds the signed distance and hole fill isosurfaces and connects them naturally where they meet.

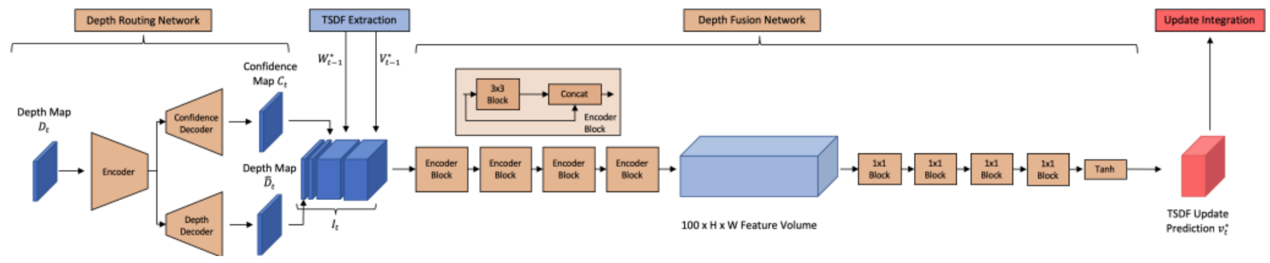
<u>RoutedFusion: Learning Real-time Depth Map Fusion</u>

- Learning based method for real-time depth map fusion. Due to its compact architecture, it requires only little training data, and is not prone to overfitting.
- Scalable and real-time capable neural architecture independent of the scene size.
- Significant improvement of standard TSDF fusion's:
    - Handles better the fusion with non-gaussian noise distributions.

  - Mitigates the surface thickening effect on thin objects and surface boundaries by avoiding inconsistent updates.

The method contains 2 network components: a depth routing network and a depth fusion network. The pipeline consists of the following four essential processing steps which are also illustrated:

- Depth Routing: from raw depth map to denoised and outlier-corrected depth map (D) and confidence (C). Also routes TSDF values along each viewing ray.
- TSDF Extraction: from routed depth to a local camera-aligned voxel grid with TSDF data (V and W) via trilinear interpolation.
- Depth Fusion: from results of the previous process (D, C, V, W) to the local TSDF update v*.
- TSDF Update Integration: transfer v* update into the global coordinate frame to get v which is into the global TSDF volumes V+1, W+1 using TSDF update equations.



As we can see, both approaches rely on D and W (the cumulative signed distance and weight functions) with also a voxel representation Moreover, they use an update mechanism until convergence.

The newest approach uses NN to predict denoised and outlier corrected depth map with also per-pixel confidence maps and also uses all TSDF data to predict the local update v*. The TSDF updates based on the data of a larger neighbourhood instead of per ray.

Conclusions

From this lab we have learned the much information can be obtained from the essential and camera matrices and that we can also obtain 3D information by using images and the configuration for those images. For the stereo matching we have seen that it is a method that has to be well implemented to not be very time expensive and that the final result is very dependent on how the parameters of disparity and window size are set. We have also seen that all these methods are subject to the different distortion that the data can have.