Jani Tähtinen

Neural network based image capture for 3D reconstruction

School of Technology and Innovations

Automation and Computer Science, Master of Science in Technology

Energy and Information technology

Vaasa 2023

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekijä: Jani Tähtinen
Tutkielman nimi: Neural network based image capture for 3D reconstruction
Tutkinto: Diplomi-insinööri
Työn ohjaaja: Jani Boutellier, Luca Ferranti
Valmistumisvuosi: 2023 **Sivumäärä:** 81

TIIVISTELMÄ:
Diplomityön tarkoitus on rakentaa neuroverkko, joka pystyy valitsemaan tärkeät kuvat 3D-mallinnusta varten videosta ilman kuvauksen tarkkuuden heikentymistä verrattuna kuvaukseen, joka on tehty kaikilla videon kuvilla. Useasti peräkkäiset kuvat videossa sisältävät samanlaista tietoa, joka ei lisää 3D-mallinnukseen tarkkuutta. Kuinka paljon kuvissa on uutta tietoa verrattuna edelliseen kuvaan, riippuu kameran liikkeestä ja liikkeen nopeudesta. 3D-mallinnuksen rakentamiseen kuluu paljon aikaa ja laskentakapasiteettia, jos kaikkia videon kuvia käytetään 3D-mallinnuksen rakentamiseen, mikä on ongelmallista sulautetuissa järjestelmissä. Tässä työssä on käytetty oletusta, että sulautettu laite kuvaisi ympäristöä ja valitsisi kuvat, joissa on tärkeää informaatiota 3D kuvauksen tekemistä varten, jonka jälkeen valitut kuvat tallennettaisiin laitteen muistiin. Itse 3D-mallinnus tehtäisiin jälkikäteen pöytätietokoneella. Työssä on tehty tietokanta neuroverkkojen opetusta varten kokonaan pöytäkoneella.
Tietokanta opetusta varten on tehty vSLAM-menetelmällä, jossa kuvista poimitaan piirteitä, joita voidaan yhdistää kuvien välillä ja niistä laskea kameran liike kuvien välillä. Jotta opetustietokantaa saadaan enemmän näytteitä, käytetyt videot on jaettu lyhyisiin kuvasarjoihin. Näin saadaan myös opetukseen käytettyä laskenta-aikaa lyhennettyä. SfM-menetelmällä on laskettu 3D-mallinnus kuvista, työssä on käytetty pistepilveä. Pistepilvet on laskettu jokaisen kuvan jälkeen. Kuva on määritelty tärkeäksi, jos sen lisääminen pistepilven laskentaan tekee pistepilvestä samanlaisemman viiste-etäisyydellä kuin pistepilvi, joka on laskettu kaikilla kuvasarjan kuvilla. Pistepilvien samanlaisuutta on mitattu viiste etäisyydellä jokaisen pistepilven laskentaan lisätyn kuvan jälkeen. Riippuen kuinka paljon viiste etäisyys pienenee kuvalle määritellään luokka. Neuroverkon rakenteena käytetään LSTM takaisinkytkeytyvää neuroverkkoa, koska se pystyy luokittelemaan jokaisen kuvan koko aikaisemman kuvajonon perusteella, eikä vain sen kuvan perusteella, jota parhaillaan luokitellaan. Matlab-ohjelmistoa on käytetty diplomityössä tietokannan ja neuroverkkojen rakentamiseen.
Diplomityön tuloksena LTSM takaisinkytkeytyvä neuroverkko pystyy valitsemaan tärkeimpiä kuvia lyhyistä kuvasarjoista, mutta kuvien valintatarkkuutta pitää vielä tulevaisuudessa parantaa ennen kuin esitettyä järjestelmää voisi käyttää sulautetussa järjestelmässä. Neuroverkko ei oppinut valitsemaan yhtä ja vain yhtä kuvaa samanlaista tietoa sisältävien kuvien joukosta työssä käytetyillä riskifunktioilla.

Avainsanat: vSLAM, SfM, LSTM neuroverkko, 3D-mallinnus, Pistepilvi

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Jani Tähtinen
Thesis title: Neural network based image capture for 3D reconstruction
Degree: Master of Science in Technology
Supervisor: Jani Boutellier, Luca Ferranti
Year of graduation: 2023 **Number of pages:** 81

ABSTRACT:
The aim of the thesis is to build a neural network, which is capable of choosing frames from a video, which have important information for building a 3D map of the depicted structure without losing the 3D map accuracy. Many times, consecutive frames have redundant information, which do not add to 3D map any significant information or some frames might be, for example, distorted, which do not add to 3D map at all. It all depends on how a camera is moved around when a video is filmed. If all the frames of the video are used in the reconstruction of the 3D map, it will take a long time and it will require a lot of resources, which is problematic especially in the embedded devices. In this thesis it has been considered that embedded device would choose the most informative frames for building the 3D map, but the 3D map itself would be built afterwards with the saved frames on a desktop computer. A database is built from video feeds for neural network training and testing.
To build the data base for training a neural network a visual simultaneous localization and mapping algorithm is used to extract features, connecting points between frames and estimate the camera movement from each frame of the video feed. To get more training samples and make the training less time consuming, video feeds have been divided into short sequences of frames. A structure from motion algorithm is used to construct a 3D point cloud of image subsets. A 3D point cloud is then constructed after each frame. To determine whether a frame is a frame with important information for 3D point cloud construction, chamfer distance is used to calculate how close the 3D point cloud is after each added frame to the 3D point cloud constructed with all the video frames. Based on the chamfer distance change then class label is determined for each frame. For the neural network a long short-term memory recurrent neural network structure was chosen, because it can learn from the entire sequence of data. The data base construction, neural network training and validation all were done with Matlab. The result of this master's thesis is a simple long short-term memory neural network that can choose the important frames from a short sequence of images, but the accuracy needs to be further improved to use the presented method in real embedded device. The custom loss function developed in the thesis did not perform well enough that any of the similar consecutive frames could be chosen, but not more than one of those.

Keywords: vSLAM, SfM, LSTM network, Chamfer distance, Point cloud

Contents

Figures

Tables

Abbreviations

$2D$	Two-dimensional space
$3D$	Three-dimensional space
AUC	Area under the curve
$BRIEF$	Binary robust independent elementary feature
CD	Chamfer distance
CNN	Convolutional neural network
$FAST$	Features from accelerated and segments test
FN	False Negative
FP	False positive
GPS	Global positioning system
GPU	Graphics processing unit
$LiDAR$	Light detection and ranging
$LSTM$	Long short-term memory
ORB	Oriented FAST and rotated BRIEF
$P3P$	Perspective-3-Point
PnP	Perspective-n-Point
ROC	Receiver operating characteristics
SfM	Structure form motion
$SLAM$	Simultaneous localization and mapping
TN	True negative

TP	True positive
$vSLAM$	Visual simultaneous localization and mapping

1 Introduction

Three-dimensional structure recognizing around our environment is very easy for us humans. A lot of effort has been used in computer vision to develop techniques that could recognize three-dimensional shapes and presence of objects in images in the way humans do as descried by Szeliski (2021). The aim has been to achieve the same level of detail with a computer as would a human get by looking around. Mathematical techniques have been developed for retrieving 3D objects from 2D images according to Szeliski (2021). Deep neural networks have been shown to have great potential in computer vision problems like in robust and real-time monocular image-based localisation (Kendall & Cipolla, 2017; Uy, Pham, Hua, Nguyen, & Yeung, 2019).

The aim of this thesis is to find a supervised classification neural network structure which is capable of choosing smallest number of frames from a video feed. The chosen frames have an important information for building a 3D point cloud of a structure of interest with minimal lost in the 3D point cloud accuracy. Based on these selected most informative frames 3D point cloud is then constructed afterwards. According to Chen, Yan, and Lin (2020) choosing of the most important frames has been researched extensively. Many of the methods proposed either consume a lot of time or are lacking in flexibility.

In addition, if all the frames of a video are used to construct of a 3D point cloud, it will take a long time and it requires a lot of calculation resources. Many times, consecutive frames have redundant information, which do not add anything to 3D point cloud. It can also be that, for example, 10 consecutive frames have very similar information, for example, a camera has not moved at all. All depends how a camera is moved around when a video is filmed. Embedded system with a camera, especially mobile ones, benefit from higher frame rates when movement is changing rapidly. When the change is smaller, smaller frame rate is enough to produce the required accuracy for the platform (Chen et al., 2020).

A neural network will be used to select the essential frames or images that have important information, so that the 3D point cloud can be built without losing any significant information while optimizing the required resources. The neural network should be able to determine, based on frames' camera movement and matching points, which of the frames to be saved and which not. Based on the saved frames, structure from motion (SfM) algorithm is used to construct a 3D point cloud. Several 3D volume information representation formats have been studied as stated in da Silva Cruz et al. (2019). These formats include light fields, holography, and points clouds (da Silva Cruz et al., 2019). Point clouds have become a first choice of 3D data representation format in environment mapping in autonomous driving systems, urban landscape mapping and virtual and augmented reality based on da Silva Cruz et al. (2019).

For a neural network to be able to select frames with important information, it needs training data. The training data is constructed by first extracting features from images. Next these features are compared to each other, and a comparison is made to find out the matching points between frames. Based on these matching points, camera movement is estimated. The visual simultaneous localization and mapping (vSLAM) algorithm is used to find out the matching points between consecutive images and to estimate the camera movement which will be inputs for a neural network. The training data is constructed to mimic that the vSLAM and trained neural network would work in an embedded system with limited resources. Limited computational resources make challenges in the real time system to get accurate, stable and well-distributed matches from images that have noise, changing lighting conditions and movement for vSLAM (Chen et al., 2020). In embedded systems, it is important to have the most informative frames to be picked efficiently to obtain robust tracking (Chen et al., 2020).

In the thesis, the training and the testing of the neural network is done in MATLAB. The data for a neural network teaching will be also constructed in MATLAB. The problem will be considered as a classification issue. The training data will be constructed so that it will have labels to tell whether construction of the 3D map is acceptable or not. Based on the chamfer distance between point cloud the label is assigned to that set of images.

The datasets of images will be constructed with different number of images. During the training, data constructed from sequential image data will be compared to structures build with all the frames. Network's ability to choose an image from image sets is also investigated with different training data. Own loss function is investigated for neural network.

At first, in the thesis are introduced methods used to build the database for training and testing and structure of neural network used. Next chapter will look into the previous studies made about choosing frames from video feed, loss functions and neural network structure. Fourth chapter presents the way the neural network would be used. The database construction with vSLAM, SfM and chamfer distance is presented in fifth chapter. Chapter six shows the experiments done with neural network structures and loss functions. The seventh and eighth chapters present the results and conclusions of the thesis.

2 Methods used in thesis

In computer vision, one of the central topics has been, right from the beginning, to recover a three dimensional structure of the environment from images according to Szeliski (2021). This is also called 3D reconstruction, where shape is derived from 2D images. 3D reconstruction can be then used to understand overall surroundings. Prince (2012) gives examples of a 3D reconstruction use in a robot that can use 3D information of its own environment to navigate in it or 3D reconstruction can also be used in object recognition. Full visual reconstruction is very challenging because the solution is non-unique based on (Prince, 2012) as for example if the light source intensity changes and the object reflectance changes as well comparably, but in opposite direction, the image will remain the same. In 3D reconstruction, the environment in question is modelled by a set of 3D points called a point cloud.

In the thesis, training database is constructed for neural networks for 3D reconstruction. From images, features are extracted, and they are matched by visual simultaneous localization and mapping algorithm. Camera movement also is extracted with vSLAM. Point clouds for 3D reconstruction are calculated from matched points between images using structure from motion algorithm. Point clouds are compared using chamfer distance. A long short-term memory (LSTM) neural network will be trained to select the essential images from image sequence that have important information so that the 3D point cloud from those selected can be built without losing any significant information.

A short description of theories related to the thesis subjects is presented related to previous descriptions of the thesis structure. First, camera model terminology used in the thesis is described. Then point clouds and chamfer distance are reviewed. vSLAM algorithm is descried in general level in following chapters. A neural network structure presented together with loss function and validation metrics used. A SfM algorithm is descried next in general level. Finally, tool used to build the neural networks and databases for its training and testing is presented.

2.1 Camera model

The camera model used in the thesis is derived from simple pinhole camera model, which is a mathematical geometric camera model that describes how 3D points are projected into an image (Prince, 2012). A pinhole camera is made of a closed chamber, which has a small round hole called a pinhole. An object situated in front of the pinhole is projected through the pinhole on the back of the closed chamber called an image plane. In the mathematical equivalent model of the pinhole camera, it is considered a virtual image that is right-way-up, Figure 1.

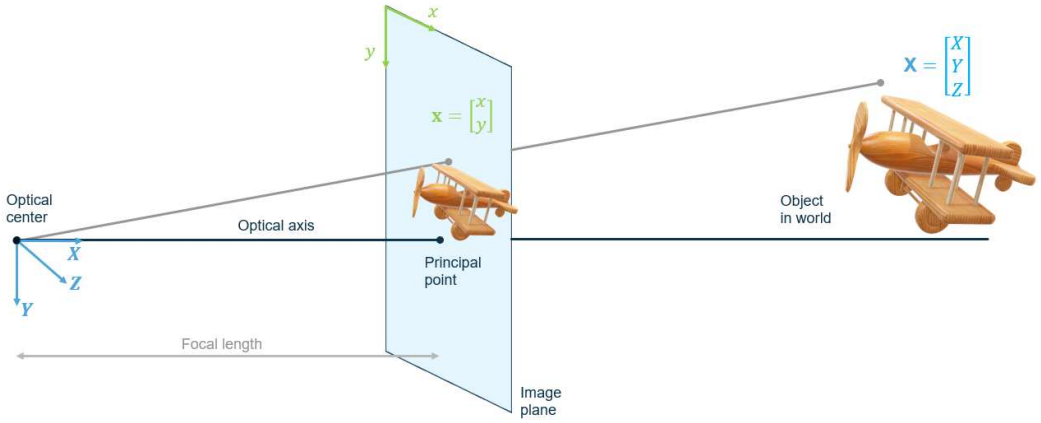

Figure 1. Pinhole camera model terminology.

The camera model expresses how a 3D point in world coordinate system is represented in a 2D image coordinate system. In the camera model, in Equation 1, a point of $[X, Y, Z]$ in the 3D Euclidean space is transferred in the 4D homogeneous coordinate presentation of $[X, Y, Z, 1]$. By using Cartesian coordinates, the mapping of the 3D world points to 2D image plane points is nonlinear. By using homogeneous coordinates, the mapping can be made to be linear. A point in the image plane also is presented in the homogeneous coordinates, in 3D homogeneous coordinate presentation of $[x, y, 1]$.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

The 3D point of $[X, Y, Z]$ is transferred to 2D point $[x, y]$ with intrinsic and extrinsic parameters. The intrinsic parameters ($\phi_x, \phi_y, \gamma, \delta_x, \delta_y$) represent the camera and the extrinsic parameters $[R|t]$ define the camera pose, the position and orientation of the camera in the world coordinates system (Prince, 2012). To transfer world coordinates to camera coordinates, extrinsic matrix is used, Equation 2.

$$[R|t] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The extrinsic matrix consists of the rotation matrix R, Equation 3, and the translation vector t, Equation 4. Rotation matrix is a combination of yaw (rotation of z axis), pitch (rotation of y axis) and roll (rotation of x axis).

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4}$$

The intrinsic matrix, shown in Equation 5, is used to convert points from the camera coordinate system to the image coordinate system. In the intrinsic matrix ϕ_x and ϕ_y are scaling factors which take in to account that an image is measured in pixels and not in physical units and the spacing of the light responding cells may be different in x- and y-axis direction. These scaling factors are calculated with the camera focal length F (Figure 1), unit is usually in millimeters, and number of pixels in x and y direction, respectively. The camera might not be in the centre of the origin of the world coordinate system. This is compensated with the principal point of the camera in pixels in x and y direction, δ_x and δ_y. With skew γ projected x is balanced to world height Y.

$$K = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

In this work, only images taken with calibrated cameras are used and therefore camera intrinsic matrices are known. If they would not be known, the camera calibration should be performed. The rotation matrix R and the translation vector t are used in this thesis as inputs for a neural network. The rotation matrix and the translation vector are calculated with visual simultaneous localization and mapping.

2.2 Point cloud

Alexiou and Ebrahimi (2017) describe a point cloud as a collection of three-dimensional points in space representing of an object. Each point is defined by its position in x, y, and z coordinates. Together the points form a cloud of points. A single point can also have associated information added to it, as in some associated features based on Alexiou and Ebrahimi (2017). 3D volume information representation formats have gained interest among the industry and academia. These formats include, for example, light fields, holography and points clouds (da Silva Cruz et al., 2019; Javaheri, Brites, Pereira, & Ascenso,

2017). Point clouds can also be used in applications where data acquiring is performed in real-time according to Javaheri et al. (2017). Applications like environment mapping in autonomous driving systems, urban landscape mapping, virtual and augmented reality and applications like photogrammetry and 3D printing are driving the choice of 3D data representation format to be the point cloud (Alexiou & Ebrahimi, 2017). Point clouds can provide a detailed yet simple representation of a 3D volume information representation as stated by Javaheri et al. (2017), which makes it increasingly popular for 3D volume information representation in computer vision.

A point cloud can be created in many ways. There are instruments that produce point clouds direct like LiDAR (Feng, Hu, Ang Jr, & Lee, 2019) and they can be created from depth sensors or be extracted from images with algorithms like vSLAM and SfM, like done in the thesis. The point cloud created by LiDAR has advantages according to Feng et al. (2019) that the point cloud created by it has a higher accuracy and the absolute scale defined directly. The point clouds reconstructed from images do not have absolute scale directly available without the help of another source. They tend to have more noise, which is due to image quality issues. Point cloud format is unstructured, which makes the quality assessment challenging problem according on Javaheri et al. (2017). To compare similarity of two point clouds, chamfer distance is chosen in the thesis to be used.

2.3 Chamfer distance

The chamfer distance is a measure of how similar the two point clouds are, it was introduced by Barrow, Tenenbaum, Bolles, and Wolf (1977) as metric. It is computed by summing the squared distances between nearest neighbour correspondences of two point clouds. This is done by taking one point from point cloud to which the other point cloud is compared to. Squared distances are calculated to every point in compared point cloud from which the smallest distance is chosen. Process is repeated with all the points in the point cloud to which the other point cloud is compared to. That way minimum distance from each point is found out to the other point cloud points. Illustration of minimum

distances are shown in Figure 2. As shown in Figure 2, minimum distance of a point to other point cloud's points might not be the minimum distance in both ways. Point 1 of the purple point cloud has the minimum distance to point 2 of the green point cloud but point 2 minimum distance to the purple point cloud is to point 3 instead.

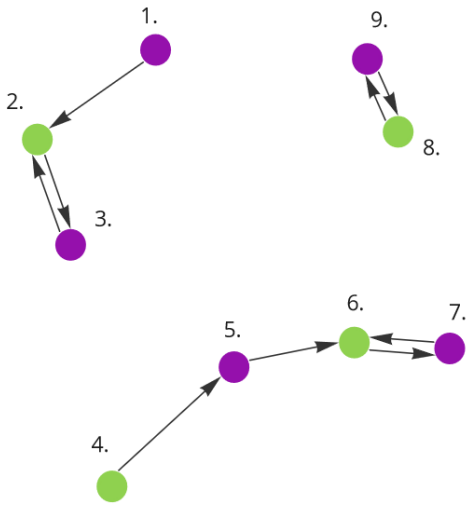

Figure 2. Chamfer distance calculation illustrated between two point clouds.

From these minima the values average is calculated, which gives the chamfer distance. If point clouds are exactly the same, the chamfer distance is zero. The chamfer distance value increases when point clouds are more different from each other. The chamfer distance is usually calculated in both ways; each point cloud is kept as reference point cloud to which the other point could is compared. In the thesis, chamfer distance is calculated in two directions. The point clouds calculated after each time image is added, is compared to the point cloud calculated from all the images. When $S_1, S_2 \in \mathbb{R}^3$ two directional chamfer distance can be calculated with equation 6. Before two point cloud are compared to each other, they need to be aligned to the same coordinate system (Luque, Carrasco, Martín, & de las Heras, 2021). It can be done, for example, with iterative closest point method presented by Besl and McKay (1992). It is not expected, that point clouds constructed from same images and with a calibrated camera, to have difference in axis

scaling.

$$d_{CD}(S_1, S_2) = \frac{1}{N} \sum_{x \in S_1} \min_{y \in S_2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_{x \in S_2} \min_{y \in S_1} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{6}$$

2.4 Visual simultaneous localization and mapping

In robotics, simultaneous localization and mapping (SLAM), introduced by Mouragnon, Lhuillier, Dhome, Dekeyser, and Sayd (2006), has been developed side by side to the computer vision community's structure of motion (Häming & Peters, 2010) according to Szeliski (2021). According to Prince (2012), a simultaneous localization and mapping is a method used to build a map of surroundings and localize item into that map at the same time. In real time environment SLAM is reconstructing a 3D model of its surroundings while moving through the environment. SLAM has been researched for many years. SLAM differs from bundle adjustment in two fundamental aspects and it allows for a variety of sensing devices, when structure from motion instead is restricted to tracked or matched feature points based on Szeliski (2021). The other one is that it solves the localization problem online, with minimal lag in providing the current sensor pose according to Szeliski (2021). This makes SLAM the method of choice for both time-critical robotics applications such as autonomous navigation and real-time augmented reality with the advantage of that it also produces better solutions with low-quality sensors (Prince, 2012; Szeliski, 2021). With extensive improvements in processing speed of computers and the availability of inexpensive sensors such as cameras and laser range finders, Prince (2012) reports that SLAM is today used for practical applications in an increasing number of areas. SLAM can use sensors like GPS, inertial measurement units, laser altimeter, LiDAR, and images. When SLAM uses images as source of information, it is called a visual simultaneous localization and mapping.

A visual simultaneous localization and mapping (vSLAM) uses images from different types of cameras and other image sensors. Cameras used with vSLAM do not need to be top of

the range cameras. Algorithms of vSLAM can be classified into two categories. According to Prince (2012), these categories are sparse methods, which match feature points of images and use algorithms such as ORB-SLAM, and dense methods, which use the overall brightness of images and use algorithms such as dense tracking and mapping. Computation of vSLAM is usually performed on a compact and low-energy embedded microprocessor that have limited processing power (Prince, 2012). Prince (2012) mentions that optimization calculations such as loop closure are high computation processes which makes the challenge of how to execute such computationally expensive processing on embedded microcomputers. Prince (2012) states, it is essential to execute image processing and point cloud matching at high frequency to achieve accurate localization with vSLAM. Visual SLAM algorithms are broadly classified into two categories, depending on how they estimate the camera motion based on Ai et al. (2021). In the indirect way feature-based method uses feature points of images to minimize the reprojection error (Ai et al., 2021). The direct way, based on Ai et al. (2021), uses the overall brightness of images to minimize the photometric error. Ai et al. (2021) continues that major part of vSLAM systems need to assume static features in the environment. The performance of a vSLAM system relying on the static-world assumption can be reduced from moving objects according to Ai et al. (2021).

SLAM is used for practical applications, but several technical challenges restrain more general adoption according to Szeliski (2021). Szeliski (2021) continues that each of the challenge has a corrective action that can help to overcome the problem. Localization errors accumulate over time causing significant deviation from actual values based on Szeliski (2021). SLAM estimates sequential movement, which include some margin of error, can also cause the map data to distort which leads to consecutive searches challenge describes Szeliski (2021). As the error accumulates, embedded device's start and stop points no longer match, according to Szeliski (2021), this is called a loop closure problem. Szeliski (2021) states that pose estimation errors like these are inevitable. Mur-Artal, Montiel, and Tardós (2015); Szeliski (2021) emphasize that it is important to detect loop closures and quantify the accumulated error. One way to mitigate this is to recall some attributes from the place which was visited before and use those to minimize the local-

ization error according to Szeliski (2021). He also states that pose graphs are created to minimize the errors. Generated map data is more accurate when error minimization is addressed as an optimization problem based on Szeliski (2021). This kind of optimization is called bundle adjustment in vSLAM (Szeliski, 2021). Localization inaccuracy can be mitigated either by using a recovery algorithm or by using the sensor data from multiple sensors to calculate the motion model (Szeliski, 2021).

vSLAM workflow consists of following steps according to (Szeliski, 2021). At first, features are extracted from the images and correspondences between images are located. Camera pose is extracted. The initialization of the map is done to find the starting point for the map. Next, features are tracked for each new frame and camera pose is estimated. A local 3D map is created with new of points. Finally, loop closure and drift correction are performed. In the thesis only the map initialization is used to extract features, to find correspondences between images and to get the camera movement. For features oriented fast and rotated brief features are used.

Rublee, Rabaud, Konolige, and Brandski (2011) presented a fast binary descriptor called oriented fast and rotated brief (ORB) which gives good performance and requires only low computational power. In their work, Rublee et al. (2011) introduced improvements to both FAST key point detector (Rosten & Drummond, 2006) and BRIEF descriptor (Calonder, Lepetit, Strecha, & Fua, 2010). Rublee et al. (2011) have introduced a orientation component by intensity centroid to FAST detector and a learning step to BRIEF descriptor to improve performance in rotation. ORB features can be used in real-time operation without GPU computations giving a good performance in viewpoint change based on Mur-Artal et al. (2015) study. vSLAM used in the thesis is ORB-SLAM.

Mur-Artal et al. (2015) introduce a novel monocular vSLAM system, which is based on ORB features. As shown by their study ORB-SLAM can operate in real time in large environments and many kinds of environments. ORB-SLAM uses ORB features to track, mapping, re-localization and loop closing, which makes the system more reliable and efficient. As shown by Mur-Artal et al. (2015) their system real time camera relocalization is not af-

fected by viewpoint or lighting and the system has novel initialization procedure that improves accuracy. In ORB-SLAM, redundant key frames are discarded, which improved long term performance.

In the thesis predictors for the neural networks are camera movement between images, the rotation matrix and the translation vector, and matched feature point between images. The neural networks are using them to predict the importance of the images for the 3D reconstruction.

2.5 Neural networks

Neural networks are technique in machine learning. They are commonly used with supervised, unsupervised, and reinforced learning paradigms. In this thesis, supervised learning is used. In supervised learning data, that is used to train the model that has labels or ground truths (Yan, 2021). The supervised learning is divided into regression problems whose target is to predict a continues real numbers and classification problems, which will predict discrete class labels. For example, in two class classification the binary representation can be used to express a target variable has value of 0 or 1. Neural network and, especially deep neural network, are very commonly used classifiers due to their structure that can be changed to be adapted to a specific need. Deep learning techniques are growing fast among 3D data due to computational resources getting more powerful. They have showed potential in solving object classification and segmentation of point cloud data even tough classifying a real-world object data set is still a very challenging task based to Uy et al. (2019).

The neural network structure is inspired by a brain. The brain has incredible abilities, which currently outperforms computers in many areas, like vision, speech recognition and learning according to (Alpaydin, 2014). Engineers are trying to copy how these functions are addressed in the brain and figuring out ways to implement them for computers. Even though not all the details are known, according to Alpaydin (2014), processing units

in the brain are neurons, which operate parallel and are connected to many other neurons, which is believed to be the key for its computational power. Neurons are processors of the brain although much simpler and slower than in a computer. On the other hand, there are a very large number of neurons in the brain compared to processors in a computer. In the brain, neurons are processing information and connections between neurons act as memory and both of them are active, whereas in a computer only processors are active and separated memory is passive. The artificial neural networks are interesting because it is believed that they will make better computer systems. In conventional approach to programming, a big problem is broken down to smaller tasks, which are clearly defined, for the computer to perform. The neural network approach is opposite, as Nielsen (2019) describes that in neural networks and, more generally in machine learning, computer is not told how to solve the problem rather it is figuring out the solution by learning from the data.

Alpaydin (2014) describes a single layer perceptron, which is a basic processing element and mimics loosely a brain's neuron. A single layer perception has one output and one or more inputs. It has a single layer of weights that can approximate linear functions. In Figure 3 a simple perception is shown. The perception has inputs $x_j \in \Re, i = 1, ..., d$, coming from an environment or from other perception outputs and an output y. Each input variable x has a connection weight $w_j \in \Re$.

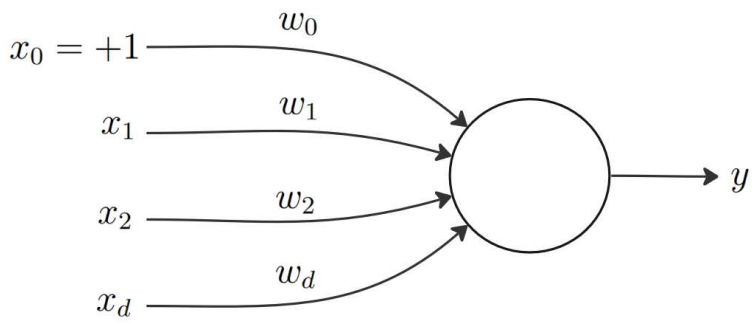

Figure 3. A simple perceptron structure.

In the simple perceptron, as shown in Figure 3, output is weighted sum of inputs as shown in Equation 7. A bias unit x_0 value is always +1. The weight coming from a bias unit is an intercept value w_0, which makes the model more general. Model weights w_j need to be learnt that correct output is generated with given inputs. Model weights are also called system parameters. If $d = 1$, it can be seen that Equation 7 becomes an equation of a line with w_j been the slope and w_0 the intercept. A single perceptron with one input and output z can model a linear line. When the input amount is increased, the line becomes a plane that can implement multivariate fit according to Alpaydin (2014). The learning of the perceptron starts with random weights. Each iteration the weights are adjusted to minimize the error between desired output and predicted output without forgetting what has been learned in previous iterations (Alpaydin, 2014).

A multilayer perceptron is a neural network structure, which is a nonparametric estimator and can be used for classification and regression problems according to Alpaydin (2014). The multilayer perceptron, also known as the feed-forward neural network, uses parametric forms for the basic functions and the parameter values are adapted during training based on Bishop (2006). For the multilayer perceptron, where there are hidden layers, Equation 7 is used to all perceptrons in that layer and are called activation a_j. When a feedforward multilayer perceptron has hidden layers between the input and the output layers, the multilayer perceptron network can be implemented in nonlinear problems.

$$a_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \tag{7}$$

$w_j 0$ are called as biases, $w_j i$ are referred as weights and a_j are called activations. Using a nonlinear differentiable function $h(\cdot)$ to transform function, activation function can have nonlinear output, shown in Equation 8. For hidden layers, a transform function is generally chosen some sigmoidal function such as the logistic sigmoid or the hyperbolic tangent function, $tanh$.

$$z_j = h(a_j) \tag{8}$$

For an output layer, previous layer values are again linearly combined to give output unit activations (Alpaydin, 2014). Shown in Equation 9 where $k = 1, ..., K$ and K is the number of outputs.

$$a_k = \sum_{j=1}^{m} w_{ki} z_i + w_{k0} \tag{9}$$

To get network outputs y_k, the output unit activations are transformed using an appropriate activation function, Equation 21, according to Alpaydin (2014). The activation function type is determined by data and assumed target variables. For regression problems $y_k = a_k$, the activation function is the identity. For classification problems, the activation function can be the sigmoid function, Equation 11, for binary classification problems and the softmax function, Equation 12, for multiclass classification problems.

$$y_k = \sigma(a_k) \tag{10}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{11}$$

$$\sigma(a) = \frac{e^a}{\sum_{j=1}^{K} e^{a_j}} \tag{12}$$

For a neural network that has one hidden layer, Figure 4, overall network function for binary classification is shown in Equation 13. In the equation (1) stands for hidden layer

and (2) for output layer. The neural network model has simple a nonlinear function, which has input variables of x_i and weights vector w give the output of y_k.

$$y_k(x, w) = \sigma\left(\sum_{j=1}^{m} w_{ki}^{(2)} h\left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{13}$$

Arrow in the Figure 4 shows the information flow direction in the network during forward propagation. According to Bishop (2006), there are many ways to name the network in Figure 4. The network could be called a 3-layer network, counting all the layers, input, hidden and output layer. It could also be called a single-hidden-layer network. Bishop (2006) suggests the terminology to call the network in Figure 4 as a two-layer network due to it having two layers that have adaptive weights, which are in a key role to determine the network properties.

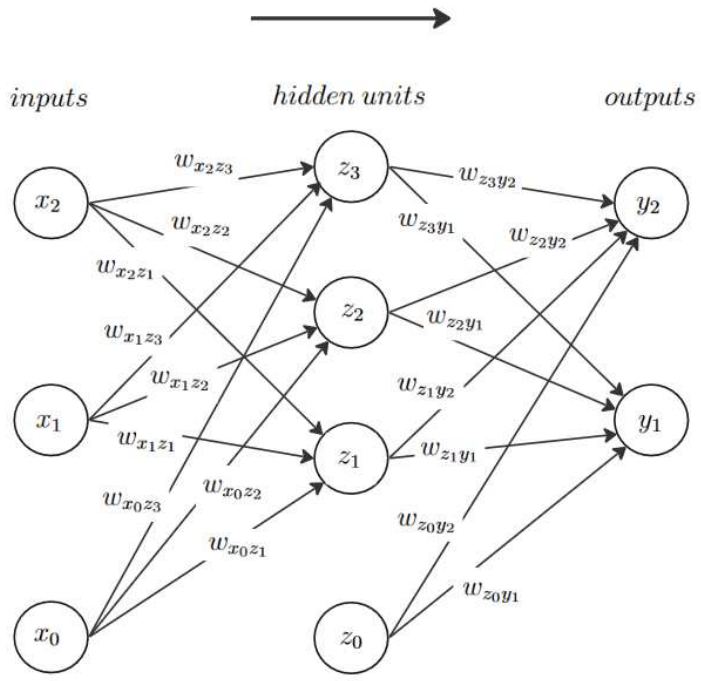

Figure 4. A neural network with one hidden layer and weights shown.

When $x_0 = 1$ is used, the bias parameters can be absorbed into the weight parameters. Then Equation 7 becomes an Equation 14 and Equation 13 becomes Equation 15.

$$a_j = \sum_{i=1}^{d} w_{ji} x_i \tag{14}$$

$$y_k(x, w) = \sigma \left(\sum_{j=1}^{m} w_{ki}^{(2)} h \left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i \right) \right) \tag{15}$$

Each of the two stages in Figure 4 appear like the single perceptron model, but perceptron model uses step-function nonlinearities, which is not differentiable. The neural network uses differentiable sigmoidal nonlinearities in the hidden units based on Alpaydin (2014), which have a key role in the network training. If error function is used to calculate the error between desired output and predicted output, it is differentiable, then gradient descent can be used as stated by Alpaydin (2014).

To determine the network parameters error between the network's predicted outputs $y_n(x_N, w)$ and corresponding ground truth values t_n is determined. Overall set of input vectors x_n, where $n = 1, .., N$ is expressed as a cross entropy error function with t_n in Equation 16, which is used for a binary classification with independent observations.

$$E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \left(t_{nk} \ln y(x_{nk}, w) + (1 - t_{nk}) \ln \left(1 - y(x_{nk}, w) \right) \right) \tag{16}$$

Based on Bishop (2006), use of cross entropy error function for a classification problem makes the training faster and improves generalization when comparing to the sum-of-squares error. According to Alpaydin (2014), the training aims to find a weight vector **w** that minimized function $E(w)$. As $E(w)$ is continuous and smooth function of w, it has smallest values at points where the gradient of the error function vanishes, Equation 17,

described by Alpaydin (2014); Bishop (2006). These points are called stationary points, which include minima, maxima, and saddle points. Based on Alpaydin (2014), it is possible to have several points in the weight space in which the gradient vanishes as the $E(w)$ typically has a highly nonlinear dependence on the weights and bias parameters. A global minimum is a point in which error function get the smallest value for any weight vector. All the other minimums are called local minimum. In neural network training, it is not generally known if found minima is global minima or not and it may not even be needed to find a sufficiently good solution according to Bishop (2006).

$$\nabla E(w) = 0 \tag{17}$$

Because it is very unlikely to find an analytical solution for the Equation 17, iterative numerical solution is used. Bishop (2006) describes the optimization of continuous nonlinear functions to be a widely studied problem and efficient ways to solve the issue can be found from literature.

In error backpropagation technique, information is sent backwards and forwards through the network in a sequence of steps to minimize the error function by changing the weights of neurons of the network (Buduma & Lacascio, 2017). The error propagates from the output to inputs, from where comes the name backpropagation based on Bishop (2006). He continues that each learning set has two different stages. In the first stage, an error function gradient with respect to the weights is evaluated. At the second stage, the calculated derivatives are used to adjust the weights. This simple technique of gradient descent is presented by Rumelhart, Hinton, and Williams (1986). Other method can also be used to calculate derivatives like the Jacobian (Loomis & Sternberg, 1990) and Hessian (Meyer, 2000) matrices in the first stage. In the second stage, the weight adjustment can be done with a variety of optimization schemes. Most of these methods are more powerful than simple gradient descent according to Bishop (2006).

In a network which has more layers, the previous layer is input for current layer and so on. Initial weights are randomly initialized. By evaluating the gradient, the direction of steepest descent can be found and a step towards biggest gradient descent direction can be taken. This is repeated to get closer the minimum error every step until the point of minimum error is found as described by Buduma and Lacascio (2017). To determine how far each step is taken, learning rate, ϵ, is used. The learning rate should not be too big that minimum error is not found, nor should it be too small that the training process takes a long time.

The backpropagation algorithm to adjust the weights of one layer in the network is shown with cross entropy error function and sigmoid activation function, a combination used in binary classification tasks (Bishop, 2006). The weights w are updated each iteration with the delta rule, Equation 18,

$$\Delta w_k = -\epsilon \frac{\partial E}{\partial w_k} \tag{18}$$

The gradient can be calculated with a chain rule shown in Equation 19.

$$\frac{\partial E}{\partial w_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z} \frac{\partial z}{\partial w_k} \tag{19}$$

A partial derivative is calculated from loss function, loss over weights. Last part of Equation 19 is partial derivative of Equation 7 over weights, which is shown in Equation 20.

$$\frac{\partial z}{\partial w_k} = x \tag{20}$$

For classification network, outputs sigmoid function is used as activation function, which

changes real number to range of $(0, 1)$. Network predictions y is shown in Equation 21. Sigmoid function is shown in Equation 11.

$$y_k = \sigma(z) \tag{21}$$

In the middle part of the chain rule, partial derivative of network prediction over activation function is calculated, shown in Equation 22.

$$\begin{aligned}
\frac{d\sigma(z)}{dz} =& \frac{d}{dz}((1+e^{-z})^{-1}) = -1(1+e^{-z})^{-2}(-e^{-z}) \\
=& \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{-z}}\frac{1+e^{-z}-1}{1+e^{-z}} \\
=& \frac{1}{1+e^{-z}}(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}) = \frac{1}{1+e^{-z}}(1 - \frac{1}{1+e^{-z}}) \\
=& \sigma(z)(1-\sigma(z)) = y_k(1-y_k)
\end{aligned} \tag{22}$$

For classification network, a binary cross-entropy loss ($E(t, y)$) is used (Good, 1952), where t is true class and y is predicted class. Binary cross-entropy loss shown in Equation 16. Binary cross-entropy loss partial derivative over prediction is shown in Equation 23.

$$\begin{aligned}
\frac{\partial E(t,y)}{\partial y} =& \frac{1}{N}\sum_{n=1}^{N} -(t_i \log y_i + (1-t_i)\log 1 - y_i) \\
=& -\frac{t}{y} + \frac{1-t}{1-y} = \frac{y-t}{y(1-y)}
\end{aligned} \tag{23}$$

To get the weights w chain Equations 19, 20, 22 23 are substituted to Equation 18, which is shown in Equation 24.

$$\Delta w_k = -\epsilon \frac{y_k - t}{y_k(1 - y_k)} y_k(1 - y_k)x = -\epsilon x(y_k - t) \tag{24}$$

Staudemeyer and Morris (2019) are describing a feed-forward neural network to be the most common type of neural network. A feed-forward neural network can only handle static classifications tasks. In classification problem, there can be two or more classes. Most classification algorithms can be reconstruct as a distance-based classifier, Alpaydin (2014). In some cases, instead of making a decision between two options' type decision, it may be needed to calculate a probability, $P(Y|X)$.

By only being able to do a mapping between input and output layers in a static way, it limits a network to be unable to do time based classification (Staudemeyer & Morris, 2019). In order a network to be a dynamic classifier, it also requires a signal from a previous time step to recur in the network. These networks are called recurrent neural networks (RNN) presented by Rumelhart et al. (1986). A simple recurrent network is able to look back in time about 10 time steps based on Staudemeyer and Morris (2019). This is since feedback signal is either vanishing or exploding. Issue of vanishing or exploding can be avoided by using a further developed recurrent neural network called Long Short-Term Memory (LSTM) recurrent neural network (Hochreiter & Schmidhuber, 1997), which is able to learn more than 1000 time steps depending from network structure as described by Staudemeyer and Morris (2019). To overcome vanishing or exploding issue of RNN, LSTM networks use additional gates to control what information is past from the hidden cell as output to the next hidden state based on Hochreiter and Schmidhuber (1997). According to Hochreiter and Schmidhuber (1997), the additional gates make the learning of the long-term relationships in the data more effectively. Hochreiter and Schmidhuber (1997) also state that LSTM networks' lower sensitivity to the time gap makes them outperform simple RNNs in sequential data analyse. LSTM block typically has a memory cell, an input gate, an output gate, and a forget gate added to the hidden state (Hochreiter & Schmidhuber, 1997). Hochreiter and Schmidhuber (1997) continue that, the weights and biases of the forget gate are used to control the amount of a value remains in the cell

and the output gate's weights and biases are used to control the output activation of the LSTM block.

A long short-term memory networks are used with sequential data because these networks can learn long-term dependencies between time steps of data (Staudemeyer & Morris, 2019). Staudemeyer and Morris (2019) mentioned applications that use LSTM networks include sentiment analysis, language modelling, speech recognition, and video analysis. Staudemeyer and Morris (2019) describe LSTM recurrent neural networks as one of the most powerful dynamic classifiers which are openly available.

Important part of the neural network is a loss function. A Loss function effects heavily how well the neural network perform in the given tasks.

2.5.1 Loss functions

Loss function is used in supervised learning to compare predicted values to ground truths. Loss function minimises the error between predicted values to ground truths depending on what the function is. Commonly cross-entropy loss is used for classification problems Bishop (2006), shown in 16. Loss function is designed to perform well in certain tasks. Loss function backpropagates error to update weights in the network. Gradient descent uses the derivatives to update the weights. The derivative of binary cross-entropy loss is shown in Equation 23. Chamfer distance loss function derivative of Equation 6 is shown in Equation 25. It was presented by Fan, Su, and Guibas (2017) for 3D object reconstruction from single image. They concluded that chamfer distance is differentiable, efficient to compute and robust against small number of outliers.

$$\begin{aligned}\frac{\partial d_{CD}(S_1, S_2))}{\partial \hat{y}} &= \frac{1}{N}[\frac{\partial}{\partial \hat{y}}(\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 + \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2)] \\ &= \frac{1}{N}[\frac{\partial}{\partial \hat{y}}((\sqrt{(y-\hat{y})^2})^2 + (\sqrt{(\hat{y}-y)^2})^2)] \\ &= \frac{1}{N}[2(y-\hat{y})(-1) + 2(\hat{y}-y)(1)] = \frac{1}{N}[2(\hat{y}-y) + 2(\hat{y}-y)] \\ &= \frac{4}{N}(\hat{y}-y)\end{aligned} \tag{25}$$

Chamfer distance loss plot is shown in Figure 5. In the Figure 5 chamfer loss and derivative of it are plotted with ground truth of zero. On the figure x-axis is chamfer distance between predicted and ground truth point clouds. When chamfer distance is loss is used with in regression problem, the derivative is shown in Figure 5. In the regression problem backpropagation correction grows as the predicted value is further away from the ground truth value.

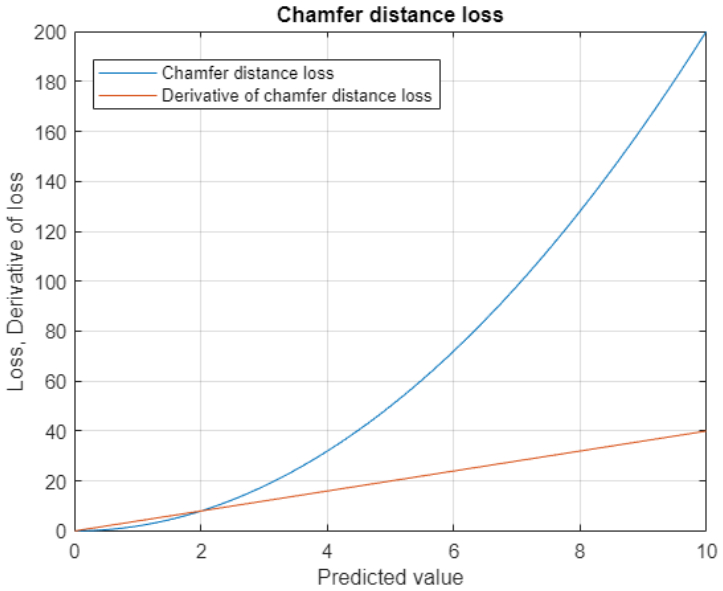

Figure 5. Chamfer distance loss in regression problem.

In binary classification problem also sigmoid function (Equation 11) has be included, be-

cause the prediction of the neural network is probability of the class. In Figure 6 is shown sigmoid function and the derivative of the sigmoid function.

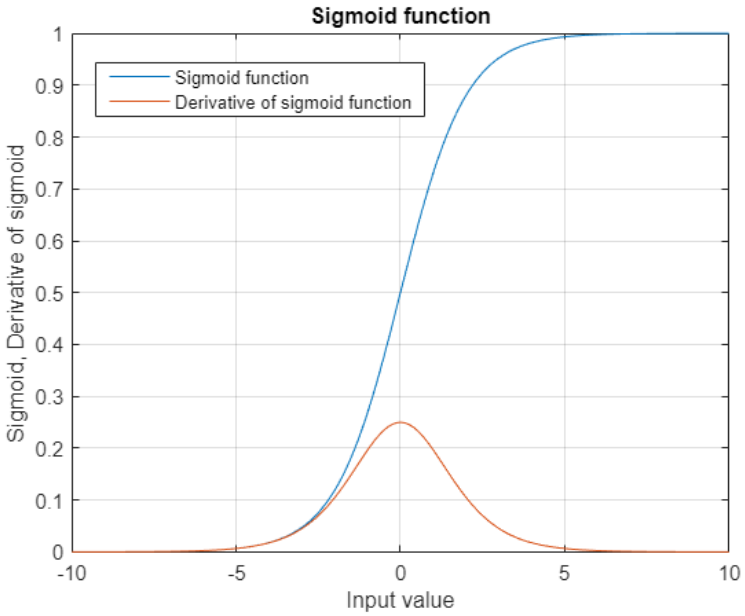

Figure 6. Sigmoid function and its derivative.

From the Equation 19 in regression problem the term $\frac{\partial y_k}{\partial z} = 1$, while in classification problem it is the sigmoid function (Bishop, 2006). The chamfer distance loss derivative in Figure 5 is calculated with $\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z} = \frac{\partial E}{\partial y_k} \cdot 1$ as in regression problem. When loss function derivative is plotted in binary classification problem, it is a product of Equation 25 as $\frac{\partial E}{\partial y_k}$ and Equation 22 as $\frac{\partial y_k}{\partial z}$ (function also shown in Figure 6) when chamfer distance loss is used and a product of 23 and Equation 22 when binary cross-entropy loss is used. Both of these loss functions combined derivative with sigmoid function are shown in Figure 7, binary cross-entropy loss on left and chamfer distance loss in binary classification problem on right. From Figure 7 can be seen that with chamfer distance loss neural network might not learning properly.

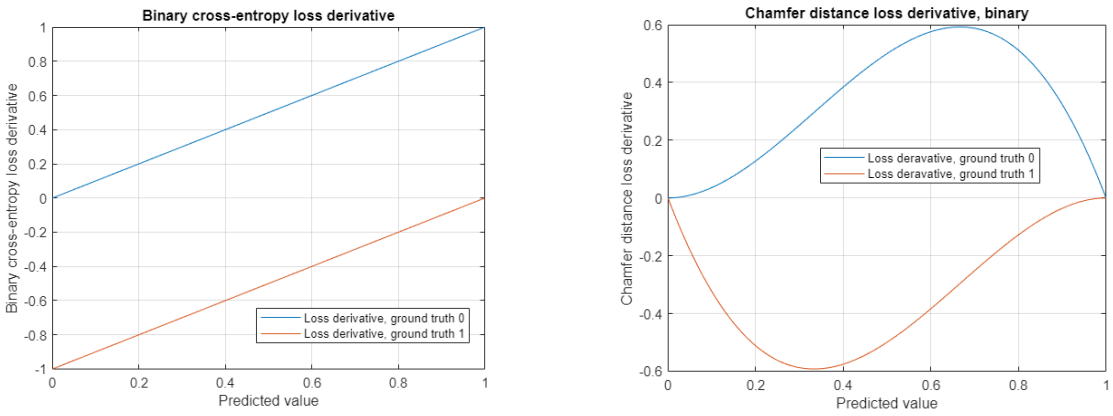

Figure 7. Binary cross entropy loss and chamfer distance loss derivatives with ground truths of zero (blue line) and one (orange line).

In Table 1 the difference of cross-entropy loss and chamfer distance loss are shown in binary classification problem with different ground truth and predicted value combinations. There can be seen that with cross-entropy loss if correct value is predicted then backpropagation correction is zero and if incorrect value is predicted there are corrections with direction. With chamfer distance loss if correct value is predicted then backpropagation correction is zero, but if incorrect value is predicted also then backpropagation correction is zero. In Table 1 backpropagation values are indicative of direction and relative magnitude of different ground truth and predicted value combinations, actual backpropagation values are real numbers.

Table 1. Binary cross entropy loss and chamfer distance loss with indicative backpropagation corrections of different ground truth and predicted value combinations.

Ground truth	Predicted	Binary cross entropy loss backpropagation	Chamfer distance loss backpropagation
1	1	0	0
1	0	-1	0
0	1	1	0
0	0	0	0

The neural network needs to be trained for the given task, for it to be able to predict reliable.

2.5.2 Neural network training

In machine learning, algorithms are developed to improve with the training based on Staudemeyer and Morris (2019). They continue that, in theory, more the algorithms are trained, better they can perform tasks. Based on the training data characteristics, same machine learning algorithm can excel in different tasks. An algorithm performance is measured by applying algorithm with some new data for the algorithm.

The error on the validation set starts to increase when more training epochs are made while the error on the training set decreases (Alpaydin, 2014). Alpaydin (2014) continues that initially all the weights are close to zero meaning that they have a little effect. The value of the most important weights will increase as training progress based on Alpaydin (2014). Almost all weights are updated away from zero if training is continued to get less error on the training set according to Alpaydin (2014). New parameters are added to the system as training continues, based to Alpaydin (2014), the system will come more complex which leads to poor generalization of the system. Learning should be stopped in the right point to stop avoid of overtraining Alpaydin (2014); Staudemeyer and Morris (2019) guide. The optimal point to stop training and the optimal number of hidden units is determined with cross-validation, the network's performance is tested during the training with validation data with data that is not used to network training (Alpaydin, 2014). Gradient descent converges to the nearest minimum which is problematic as the error function has many minima due to the nonlinearity of the error function according to Alpaydin (2014). (Alpaydin, 2014) guides to train the same network several times with different initial weight values to define expected error, and compute the average of the validation error. Once training is done, the results need to be validated to describe how well the network performed.

2.5.3 Validating results

Neural network predictions should be evaluated to determine how good the predictions are. In binary classification, problem predictions can be divided into four cases, true positive (TP), false positive (FP), true negative (TN) and false negative (FN) (Alpaydin, 2014) which can be presented in confusion matrix. True positive case is when ground truth is positive, and the prediction is also positive. If the prediction is positive, but ground truth is false, it is called a false positive. True negative case is when both ground truth and prediction are false. In false negative case, the prediction is false when ground truth is true.

From these four cases several metrics can be calculated. Overall prediction accuracy can be calculated as shown in Equation 26. Precision shows how well model predicts positive labels, Equation 27. With perfect precision value, all positive predictions made by the model are predicted correctly, but not all positive values are necessary predicted correctly. Recall, also known as sensitivity, describes how well model identified true values correctly, Equation 28. All positive values are predicted correctly, but also some negative values could have been predicted to be positive. F1-score is combination of precision and recall shown in Equation 29. F1-score describes model overall performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

$$Precision = \frac{TP}{TP + FP} \tag{27}$$

$$Recall = \frac{TP}{TP + FN} \tag{28}$$

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{29}$$

The classification performance can also be compared with receiver operating characteristic curve (ROC), which shows model performance with all classification thresholds. With changing the threshold, the model can be tuned to classify more positive or negative values depending on the need. ROC shows whether precision or recall is more dominant based on Alpaydin (2014). Area under the curve (AUC) value is calculated from the area under the ROC curve, which gives model performance over all possible thresholds as presented by Yan (2021). When AUC value is one, the model has a good measure of separability. Poor model would get the AUC value of zero.

Once the neural network has predicted which images should be included in the 3D reconstruction, point clouds are calculated from them with structure from motion.

2.6 Structure from motion

One technique for establishing correspondences between 2D images and to build a 3D model of a scene and to get the camera position with respect to object is structure from motion presented by Häming and Peters (2010). It involves simultaneously estimating both 3D geometry, also called structure, and camera pose, in other words, motion (Szeliski, 2021). Structure from motion is the process of reconstructing 3D structure from projections of a series of 2D images taken from different viewpoints according to Szeliski (2021). Incremental structure from motion is a sequential process that commonly starts with feature extraction and matching, followed by geometric verification (Szeliski, 2021). The reconstruction stage is constructed from the resulting scene graph, build from selected two-view reconstruction, before adding new images, triangulating scene points, filtering outliers, and refining the reconstruction using bundle adjustment according to Schönberger and Frahm (2016)

A special case of feature-based positioning is estimating an object's 3D pose from a set of 2D point projections based on Szeliski (2021). In this pose estimation problem the position and orientation of the object in its coordinate system is describe in the camera coordinate system (Szeliski, 2021). To recover the pose, it needs at least three correspondences, if the camera is already calibrated, and is known as the perspective-3-point-problem (P3P), with larger numbers of points generally known as PnP according to Szeliski (2021).

In triangulation a point's 3D position is determined from a set of corresponding image locations and known camera positions (Szeliski, 2021). Szeliski (2021) descries the easiest way to solve this problem is to find the 3D point that lies closest to all of the 3D rays corresponding to the 2D matching feature locations.

Bundle adjustment is the most accurate way to recover structure and motion by performing non-linear minimization of the measurement errors based on Szeliski (2021). Szeliski (2021) states that it is now the standard method of choice for most structure from motion problems. Solving bundle adjustment every iteration becomes impractical when the problem grows bigger due to high computational capacity need according to Szeliski (2021). And because of the long calculation time, Szeliski (2021) reports that structure of motion is not suitable for real time environment.

The first step of global structure of motion is to remove outliers by removing incorrect point correspondences to solve the relative orientations from pairwise matching between different views, many outliers still exist because the epipolar constraints among local views usually do not reveal outliers in long point tracks according to Wen, Zou, Miao, Ying, and Liu (2019). Outliers will reduce the structure estimation if they are not handled properly (Wen et al., 2019). Many computer vision applications outlier removal in model fitting is fundamentally critical, including fundamental matrix estimation, homograph matrix estimation, vision-based robotics navigation and global outlier removal according to Wen et al. (2019).

2.7 Tools

Many platforms exist in which deep learning algorithms can be used. There are open-source libraries and frameworks for example for Python available. MATLAB is a computing environment and proprietary programming language developed by MathWorks (The MathWorks, 2022). MATLAB uses toolboxes for neutral network construction and solving. MATLAB, (The MathWorks, 2022), has also toolboxes for computer vision, especially for autonomous vehicles, visual object detection, semantic segmentation, and digital image processing. MATLAB enables use of multiple GPUs, parallel computing, cluster computing, cloud computing for accelerating deep learning calculations, (Yan, 2021). MATLAB can be used in a desktop version as well as an online version.

3 Previous work

Previous work on selecting important frames from video feed, point cloud usage in neural networks, neural network and different loss function options are reviewed. The works done by others will be compared to the results achieved in this thesis.

3.1 Selecting important frames

Selecting important frames with a fixed time or frame interval with parallel tracking and mapping method has been investigated by Klein and Murray (2007). A high-quality tracking is required and key frames need to meet an exact transformation and rotation angle. Tan, Liu, Dong, Zhang, and Bao (2013) used image overlap to choose key frames, but to name a frame as key frame it has to meet several conditions. The camera position and attitude must be estimated successfully, number of extracted feature points should be over a specific threshold and shared feature points with last key frame should be lower than a specific threshold value. Zhuang, Rui, Huang, and Mehrotra (1998) showed that key frame selection, according to the image content index by calculating feature clustering space of the current frames and checking that feature distance is over a threshold, is highly efficient way but lacks in accuracy. In their study Qin, Li, and Shen (2018) choose key frames according to parallax that the average parallax of tracking feature exceeds a certain threshold and the number of tracking features is less than another threshold. Mur-Artal et al. (2015) have adopted the key frame selection of the fittest and ability to remove later redundant frames with ORB-SLAM to achieve robustness in difficult scenarios without additional calculation cost.

Key frame choosing from a real-time high frame rate camera for vSLAM is investigated by Chen et al. (2020) who are showing that their automatic key-frame choosing method based on the improved PWC-Net for mobile platforms is improving the pose tracking quality and increasing robustness against error caused by dynamic blur. PWC-Net was introduced by Sun, Yang, Liu, and Kautz (2018), which is a compact and effective CNN

model for optical flow, a technique used to describe image motion. PWC-Net is constructed with pyramidal processing, warping, and a use of a cost volume. According to Sun et al. (2018), it outperforms all published optical flow methods. Chen et al. (2020) are using two step decomposition to get the change of posture angle between frames in their work. If threshold is not exceeded between images, then key frames will be selected in certain intervals (Sun et al., 2018). When threshold is exceeded, PWC-net is used to associate the image content of the corresponding front and back frames. Sun et al. (2018) method uses pre-trained neural network to extract contextual pixel information from input frames. Optical flow algorithm is used to estimate bidirectional flow between input frames according to Sun et al. (2018). In their system Chen et al. (2020) input frames and their context maps are fed to a video frame synthesis neural network to produce the interpolated frame in a context-aware fashion. If the change between frames exceeds the threshold, then the improved PWC-Net is used to associate the image content of the corresponding earlier and later frames to make the intermediate frame as a key frame (Chen et al., 2020). With this work flow Chen et al. (2020) achieves key frames to be more focused in complex areas and less in flat areas.

3.2 Point clouds

Deep learning techniques have showed potential in solving object classification and segmentation of point cloud data even tough classifying a real-world object data set is still a very challenging task according to Uy et al. (2019). Feng et al. (2019) have developed the 2D3D-MatchNet, which is an end to end triplet-like deep neural network architecture that learns the descriptors from 2D images and key points from 3D point clouds. Feng et al. (2019) are using 2D3D-MatchNet for visual pose estimation with directly matching 2D to 3D correspondence of images and point clouds. In visual pose estimation, the camera pose is estimated in respect to coordinate frame of a reference point cloud (Feng et al., 2019). The visual pose estimation is important part of visual Simultaneous Localization and Mapping (Mur-Artal et al., 2015) and structure from motion (Schönberger & Frahm, 2016). 2D3D-MatchNet localization results are better than ORB-SLAM2 (Mur-Artal

& Tardós, 2017) in a large scale outdoor urban environment. DeepI2P is deep neural network used to cross-modality registration of 2D image and 3D point cloud developed by Li and Lee (2021). It classifies whether the projection of each point in the point is in front of the camera or behind it (Li & Lee, 2021). Labelled points are then directed into an inverse camera projection solver to estimate the relative pose (Li & Lee, 2021).

3.3 Neural networks

Hoang, Lilienthal, and Stoyanov (2020) have used a combination of convolutional neural networks and dense simultaneous localization and mapping system to recognize objects and recover camera pose. Ai et al. (2021) have combined ORB-SLAM2 (Mur-Artal & Tardós, 2017) with object detection by deep neural network to reduce dynamic content influence. In their system, deep object detection neural network is pre-processing data to dynamic targets or static objects. With this Ai et al. (2021) are achieving more robust and accurate localization and mapping results in highly dynamic scenarios.

3.4 Loss function for neural network

Kendall and Cipolla (2017) have used combined loss function to learn camera pose using PoseNet (Kendall, Grimes, & Cipolla, 2015) deep learning model. To combine localisation metrics to the same loss function they used a linear weighted sum $L_\beta(I) = L_x(I) + \beta L_q(I)$ of position and orientation errors to approximately equal. Camera position and orientation are expressed in different units, a scaling factor β is needed. Kendall and Cipolla (2017) found that β factor requires significant tuning to get reasonable results, but model that learns together the camera's position and orientation, performs better than a model that has been trained the position and the orientation separately. Similar combined loss function is used by Uy et al. (2019). They are using deep neural network to real-world point cloud classification with crowded backgrounds. Their loss function combines cross entropy losses of classification and segmentation by setting fac-

tor to 0.5. Segmentation loss is used to separate foreground points from background points. Feng et al. (2019) are using in their 2D3D-MatchNet the weighted soft-margin triplet loss $L = \ln \alpha d$ introduced by Hu, Feng, Nguyen, and Lee (2018), which combines image and point cloud descriptors. They use Euclidean distance of $d_{positive}$ and $d_{negative}$ as d. Their deep network can converge faster and increase the retrieval accuracy thus achieving better localization result. Wang, Ang, and Lee (2021) used chamfer distance to calculate the reconstruction loss, which can be used for big data sizes as well. Their overall loss included the reconstruction loss between the generated point cloud and the ground truth, the adversarial loss, and the feature matching loss between the partial and complete point features. Chamfer distance loss was presented by Fan et al. (2017) for 3D object reconstruction from single image. They concluded that chamfer distance is differentiable, efficient to compute and robust against small number of outliers. Ping, Esfahani, Jiaying, and Han (2022) have identified that Chamfer distance loss will assign an equal weights to all points inside the point clouds. Chamfer distance loss is used by Nazir, Afzal, Pagani, Liwicki, and Stricker (2021) to classify and shape completion of point clouds and Saabni and El-Sana (2011) have used modified chamfer distance loss for spotting handwritten words.

4 Scheme structure

In the thesis, it is consider an environment where the developed neural network would work, shown in Figure 8 on left. It is considered that neural network developed in the thesis would work in embedded device, which has limited calculation resources. The embedded platform would have a video camera, which would take frames, for example, in a rate of 50 frames per second. Each frame is fed into a vSLAM calculation, and from the calculation the input data for neural network is received. Trained neural network processes each added frame one at a time and outputs binary classification, does the frame have important information for point cloud construction. Embedded device saves the images that are considered to be important and afterwards point cloud is constructed with desktop computer with SfM.

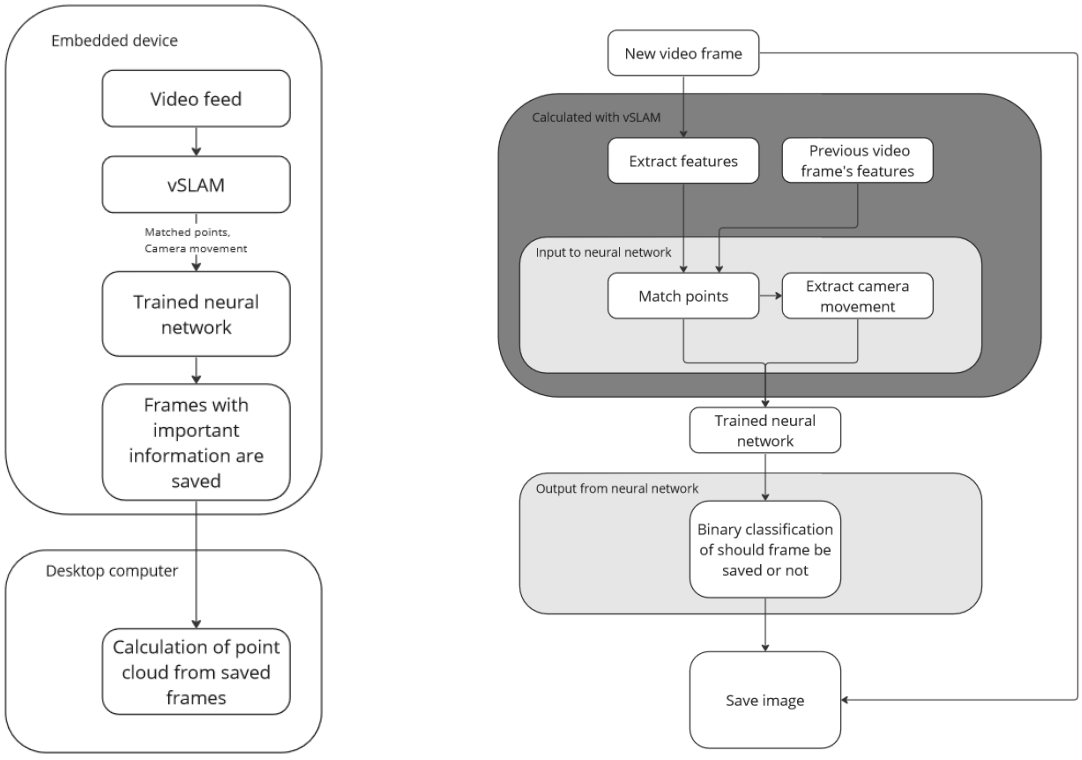

Figure 8. Environment for the developed neural network to work on in embedded device on left and process description of the system in the embedded device on right.

The assumed process in the embedded device, Figure 8, includes feeding each video frame into vSLAM calculation, which extracts features and feature points from the frame and matches the points with the previous frame's points. Camera movement is extracted from matched points. Matched points and camera movement is used as inputs to the neural network for prediction if that frame has important information and should be saved. The matched point represent how similar images are and camera movement that how much movement there has been between frames. The neural network is classification network that has binary output of not to save an image (0) and save an image (1). Based on neural networks output then image is saved to embedded devices storage space. Video frames are fed into the vSLAM calculation one by one in order. As neural network should learn changes in the frames, that is why recurrent neural network is used, which allows network to remember also previous steps and not only the current step.

The network training and testing are done following the similar process that would also be used in the embedded device, but in the thesis all the steps are done with desktop computer and using MATLAB. In the thesis a data base is constructed, which is used to train the neural network and to test it. The data base includes predictors calculated with vSLAM and ground truths for each image determined by chamfer distance change between point clouds calculated with SfM after each image and point clouds calculated from all the images. The process of training and testing the neural network is shown in Figure 9. In the thesis several ways to construct the data base are tested especially how to determine the ground truths for each of the images.

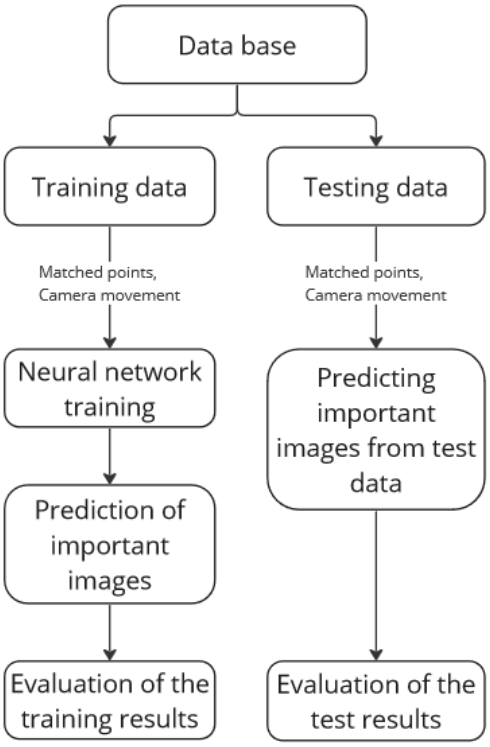

Figure 9. Training and testing process used in the thesis.

For the start, databases are created which can be used to train and test the neural networks. Several databases are made because labels, especially, change during experiments.

5 Construction of databases for training and testing

To train a neural network for some specific tasks, it needs training data. With the training data, the neural network is trained to perform specifically on similar data also in the future. The training data will have some number of predictors, which are then used to predict, in the case of classification, labels. In the training, data ground truth labels are signed for the predictors. During the training, predicted labels from the neural network are compared to ground truths and the neural network weights are adjusted based on them. The training data needs to have information how the task is generally solved.

There are a lot of labelled image sets available that can be used for a neural network training, but not suitable for the issue considered in this thesis. To train and to test a neural network, a database is constructed for the thesis. The database for neural network is constructed in MATLAB using image sets collected by Sturm, Engelhard, Endres, Burgard, and Cremers (2012). The datasets are available in the Technical University of Munich's TUM Department of Informatics Computer Vision Group (2022) web pages. The image sets used are from the category of handheld SLAM. The used sets are fr3/long_office_household, fr2/large_no_loop, fr1/360 and fr1/desk. The image sets have 2585, 3359, 756 and 613 images in each of them respectively. The image sets are taken with monocular camera and camera intrinsic parameters are known to include the camera focal length, principal point, and image size. The image sets are video streams that are converted to set of images. As the image are real video streams, they already have images, which are blurred, out of focus or images look stretched or sheared, no additional image augmentation was used.

The training data is constructed as shown in Figure 10. The training data construction starts from the chosen image sets. They are first divided into subsets to get more data for the training and for more manageable data sizes. From each image subset, matched points between images and camera movement are calculated with vSLAM. This information is used as predictors for the neural network. To get a label for each image whether

it holds key information or not, point cloud is calculated after each image. These point clouds are then compared to point cloud calculated with all the images in the image subset. Chamfer distance is used to measure how close the two point clouds are to each other's. Based on how much chamfer distance changes when image is added to point cloud calculation, a label is signed to the image. Different thresholds are used to determine how much chamfer distance must change before image is labelled as an image to be saved. In the final state, training data is saved to MATLAB data storage object, which can then be used to train a neural network.

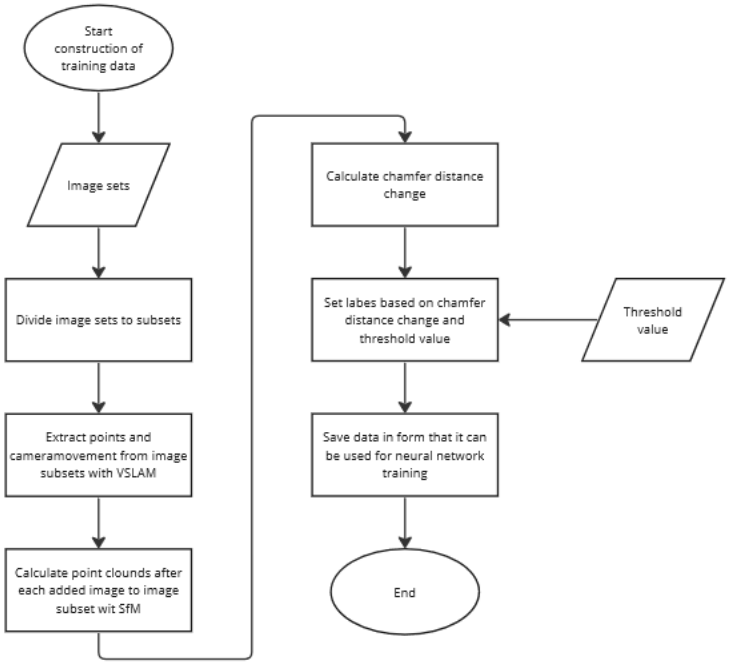

Figure 10. Flowchart of training data construction.

5.1 Image subsets for database

All the chosen image sets have too many images for the neural network teaching purposes done in this thesis. If entire images sets would have been used for training, the calculation time would have been extremely long, and many more image sets would have

been needed to have enough data for a neural network to learn. All chosen image sequences are divided into smaller parts to get more data for training and more manageable calculations times. Image sets have been divided into smaller subset of 25 images in each for sequential neural network input and subsets of 10 images from where one image is picked. Some subsets are taken directly from original image sets while others have been constructed by taking images in steps and some subsets are taken more randomly. In all cases, images in the subsets are in the same time order as in the original image set. In Figure 19 is shown how image set fr3/long_office_household has been divided into multiple subsets. All image sets are divided in a similar fashion.

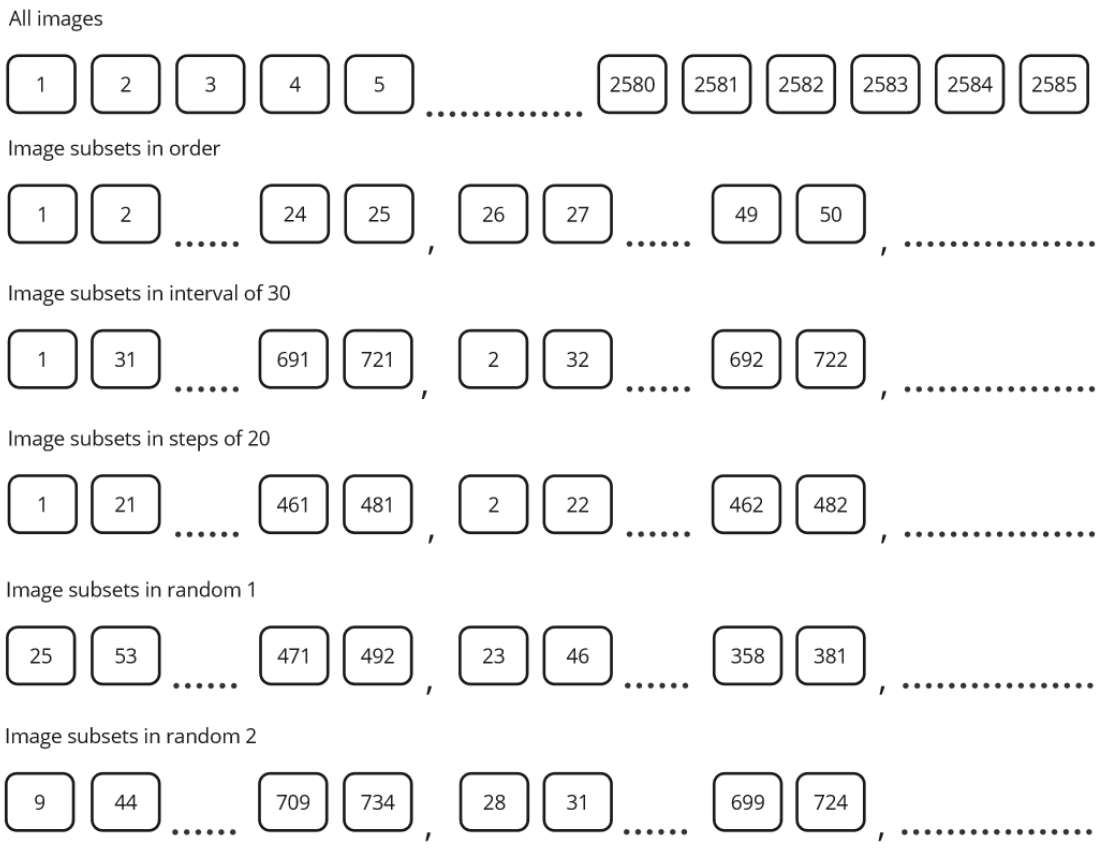

Figure 11. Dividing fr3/long_office_household image set to subsets.

At first, on the top row of the Figure 19, all the images are illustrated in the whole image

set. Image set fr3/long_office_household has the total of 2585 images. The first 103 subsets are taken in order, so that each subset has 25 images as shown in the second row of the Figure 19. By taking images in order represents normal video feed with steady and slow camera movement.

The next subsets are taken with an interval based on how many image sets of 25 will fit to total amount of images. With this image set, the interval will be 30, meaning that the first image is taken and then 31st image is taken next and so on until the subset also has the total of 25 images. There will also be 103 subsets from this method. The next subsets are taken in 20 image steps. It is very similar way than in interval subsets, but in this case, the number of images is not decided based on how many certain size subset will fit the whole image set, but rather fixed step size. With the step size of 20, there will be 20 subsets that have 25 images in each subset. Two subsets taken in images in interval or in steps represents video feed that is taken by moving camera fast, but in steady speed.

The two last ways to divide image sets to image subsets in the Figure 19 are two ways of randomly taking images from the image sets. In the random 1 way, the first image is randomly chosen from the first 25 images. When the first image is chosen then the next image is randomly chosen within the next 25 images from the first image. This will be continued until there are chosen 25 images to each subset. 15 subsets are constructed this way. In the random 2 method, the first image is chosen similarly within the first 25 images as in random 1 method. The second image is chosen randomly within the next 25 image set, meaning that an image is chosen from images 26 to 50 of the image set. The third image is chosen from images 51 to 75 of the image set and so on until image subset has 25 images. 15 subsets are also done like this. Two subsets formed with randomly represents video feed that is taken by moving camera with variable speed of the camera movement.

From the 2585 images, 256 image subsets are constructed. From all four image sets, total of 782 image subsets are constructed. All the image subsets have 25 images for those neural network structures that are trained with sequential one image information

at the time input. For a neural network training where number of images are fed to the neural network with one go as an input, image subset length of 10 images is used. The training data is otherwise build as descried above.

5.2 Predictor extraction

After the image set have been divided into subsets, the vSLAM is used to find 250 ORB-features from each of the images and match the features between consecutive images. Camera movement between images is also determined. It was decided to use matched points between two images together of the camera movement between images as predictors for the neural network training for each image. These are fast and easy to calculate, also with low calculation power. The matched points between two images describe how similar images are with each other. Camera movement is descried with rotation matrix and translation vector. There is together of 12 values describing camera movement between images. Number of matched points between images is changing between images. There can also be image pairs that do not have enough matching point to determine camera movement.

5.3 Determining label for each image

Structure of motion is used to calculate point clouds from the image subset. When images come from a video camera, they come one by one. That is modelled by calculating a point cloud after two images at the beginning of each image subset. Then a new point cloud is calculated when next image, the third image, is added. This will continue until all images are included in the point cloud calculation, shown in Figure 12.

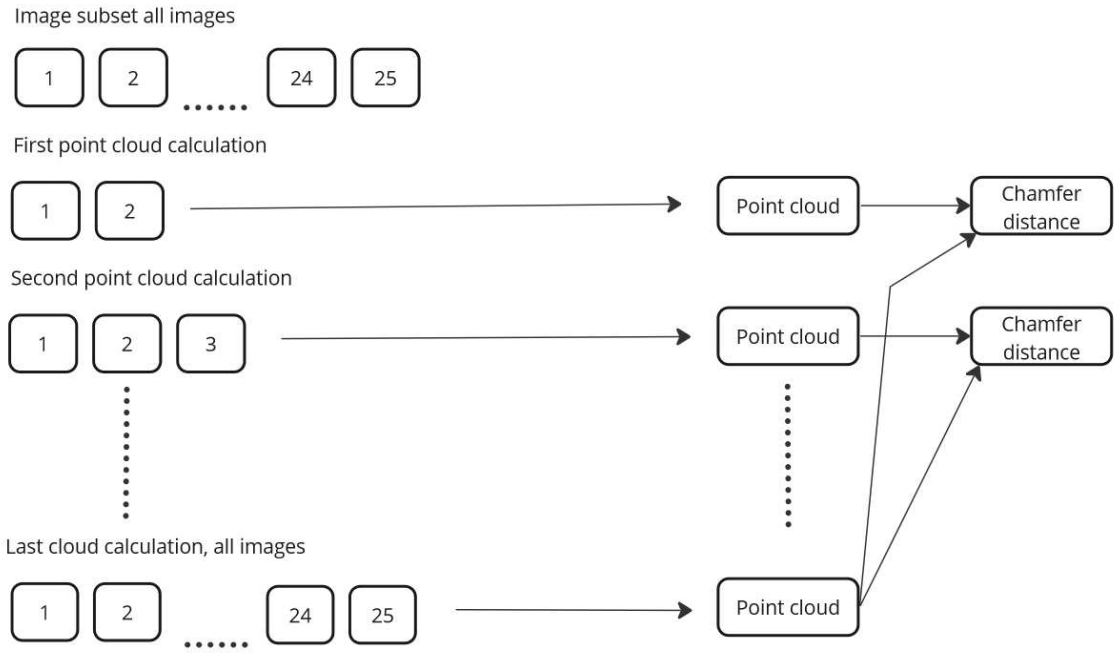

Figure 12. Process how chamfer distance is calculated in different states when images are added.

Point clouds are compared to the point cloud calculated with all the images in the subset. Similarity of the point clouds is measured with chamfer distance. Change in chamfer distance is determined to be a sign that the image has important information for the point cloud construction. Based on a threshold value, a label is signed for the image, if the chamfer distance is reduced more than a threshold value amount compared to previous chamfer distance value, show in Figure 13. If an image is not needed, false label is given and if image is needed then a true label is signed to it.

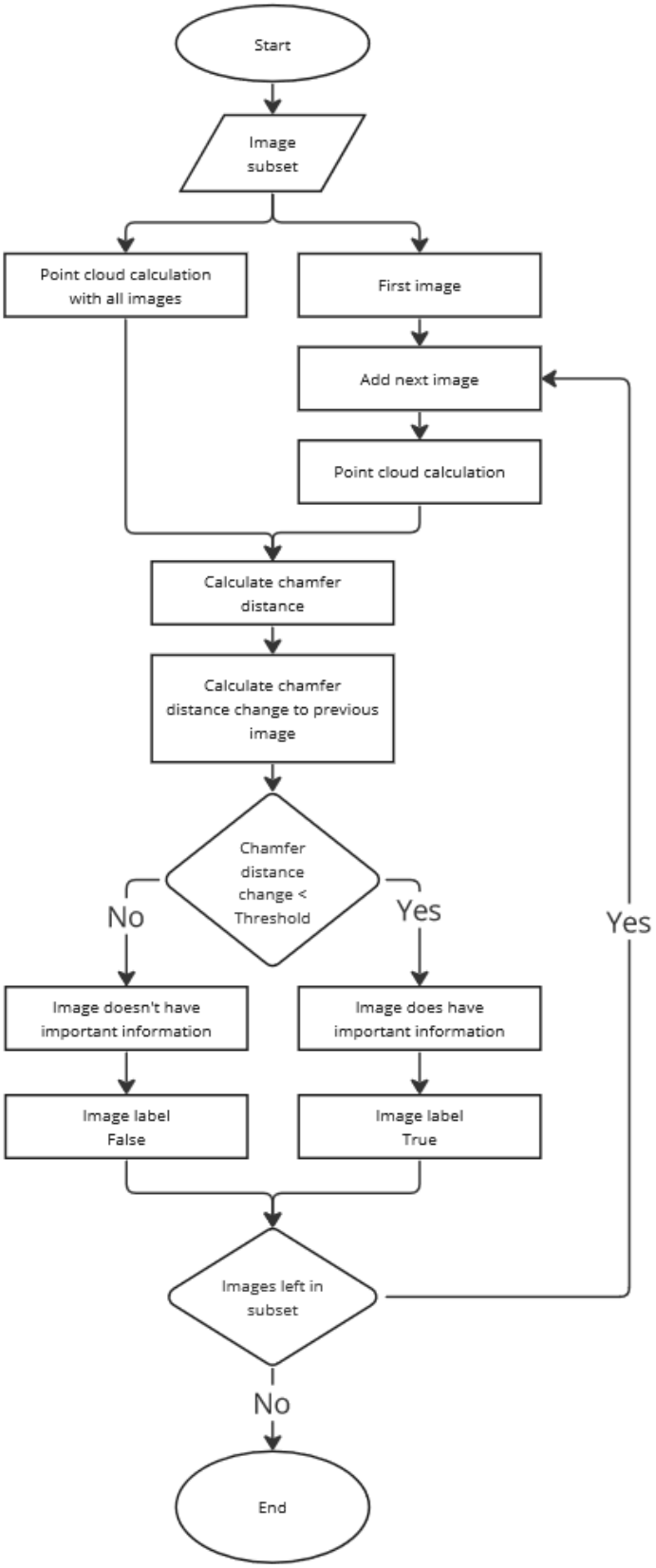

Figure 13. Flowchart of how it is determined if image has important information for a point cloud construction.

Figure 14 shows how the chamfer distance value changes after each added image with one of the image subsets. The first point cloud can be constructed from two images, from which also the first chamfer distance is calculated between the point cloud with all the images. That's why the first chamfer distance value is calculated after the second image in the Figure 14. At the beginning the chamfer distance is relative stable until the 12th image is added. After that, there is a big improvement meaning that the chamfer distance has decreased. Also, the image 16 seems to improve the chamfer distance value considerably as also does image 22. The images are labelled in the training data so that the first image of the image subset is always set to be saved. From Figure 14 can be seen that, for example, images that decrease the chamfer distance, the most are the images 1, 8, 13, 16 and 22. Using these images to calculate the point cloud, we will receive a closed point cloud to that point cloud that is calculated with all the subset images. The situation is not unambiguous because also images 2, 9, 14 and 23 give very similar decrease of chamfer distance compared to the previous image. Any of the images that have a similar chamfer distance value could be chosen to get similar results, but only one of these images should be chosen. For example, images 13, 14 and 15 have very similar chamfer distance values, so one of these images should be included in images from which the point cloud is calculated. If two images are chosen from images 13, 14 and 15, the chamfer distance is not improving, but the calculation time of the point cloud will increase.

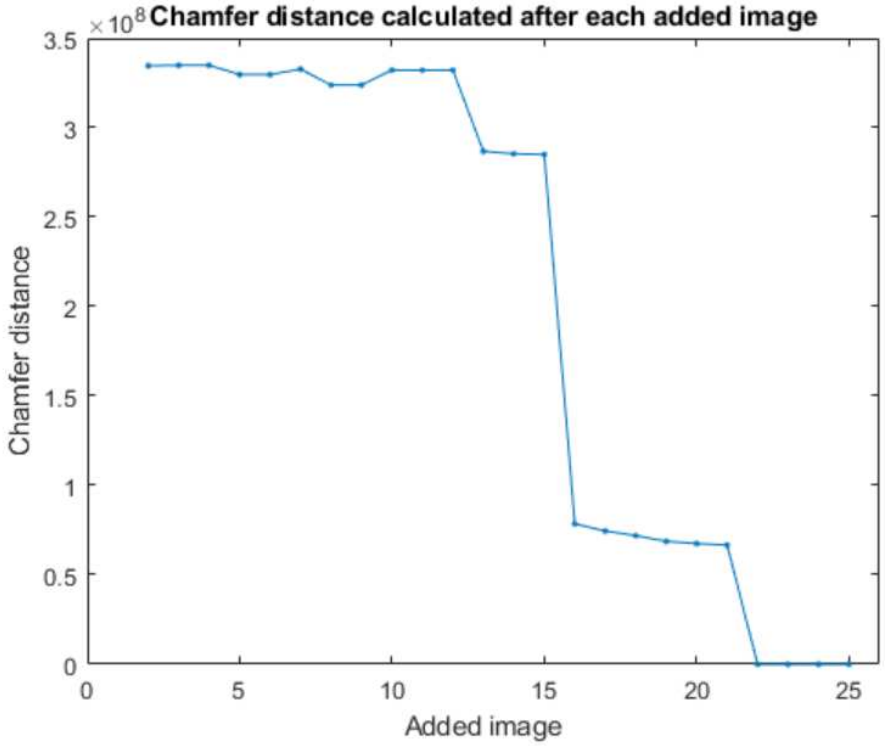

Figure 14. Chamfer distance calculation after each added image.

Once the databases are done, experimenting with different neural network structures can begin. The training data effect for the prediction performance is experimented.

6 Experiments

Constructed databases are divided into training data and testing data for the neural networks. One of the smaller image sets is chosen to be the testing data, which is not used during the network training. It is only used for the network testing. The same image set is used for testing in all the experimented cases in the thesis.

Training data labels distribution is shown in Figure 15 for chamfer distance change threshold of -5 %. Training data has 68 % of the labels zero and 32 % of one. The label zero meaning that the image should not be saved and one meaning it should be saved. As the training data is divided into image sequences of 25 images, this means that, on average, out of those 25 images network should pick 8 images, which are then used to construct the final point cloud. Often imbalance data is handled some way to avoid predictions to be biased. The minority class should also be predicted with an equal importance. In this case, resampling, over or undersampling, are not options, because there is no way to increase the images to be chosen without also increasing the number of not to be chosen or vice versa. The threshold of choosing the images to be saved can be changed. Choosing the evaluating metric to more describe both classes can be used. According to Luque, Carrasco, Martín, and de las Heras (2019), a major issue with an imbalanced data set is to determine a suitable evaluation metric. F1-score is taking into account that both classes are predicted, and it only increases if both the number and the quality of prediction improves both of the classes.

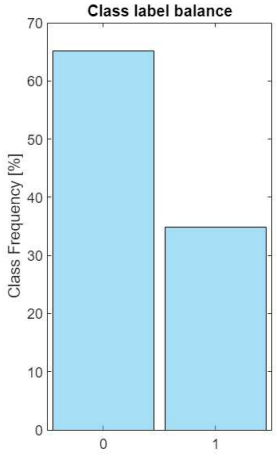

Figure 15. Balance of label classes in training data with chamfer distance threshold of -5 %.

6.1 Neural network depth comparison

Different neural network structures are tested, and multiple evaluation metrics are compared. The cross-entropy loss is used. In Figure 16, six different networks are tested with different amount of hidden units. The network structure consists of a sequence input layer, LSTM-layer with different amounts of hidden units, batch normalization layer, relu-layer, dropout layer, fully connected layer, softmax layer and classification output layer. If two LSTM layers are used, then layers from LSTM layer to dropout layer are all duplicated one after another. In MATLAB, sequence input layer is used when sequence data is used for training the network. LSTM-layer used is a bidirectional LSTM layer that learns long-term dependencies between time steps of sequence data. In LSTM layer, a hidden unit number can be specified. A bigger number of hidden units can remember more information from previous time steps, but it does not determine of how many time steps the layers remembers information. A batch normalization layer normalizes the data observations from each channel independently. It aims to stabilize the learning process and reduce training epochs needed for the network training. Relu-layer, the rectified linear activation function, will output positive input directly and zero if input is negative. A relu-layer has been found to make the training easier and many times achieve better per-

formance. A dropout layer drops nodes randomly during the training, which is effective regularization method to reduce model overfitting and improves its generalization. In a fully connected layer, inputs are multiplied with a weight matrix and a bias vector, which means that all inputs influence every outputs. In this layer, it is also defined how many outputs the network will have. A softmax layer applies a softmax function to the input of the layer. A classification layer uses the cross-entropy loss for classification.

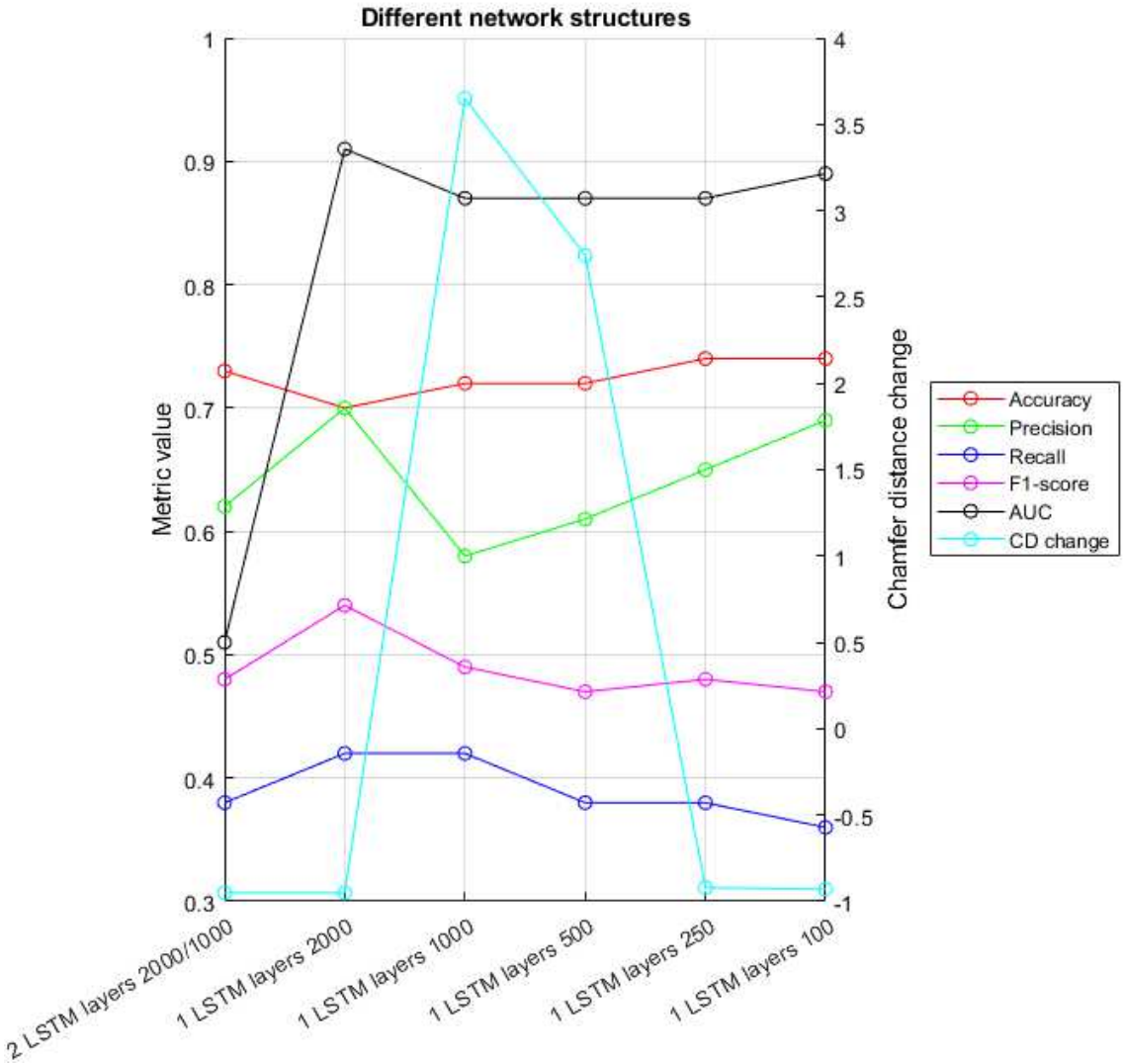

Figure 16. Evaluation of different network structures on the test data.

In Figure 16, it is shown that two LSTM-layer will cause the network to overfit and it does

not perform well with test data. With a lower number of hidden units, the model overall performance is comparable to a higher amount of hidden units. Overall accuracy does not change with different network structures. The recall that describes how well the true labels are predicted correct is very low. All the metrics, accuracy, precision, recall, F1-score, or AUC are showing better performance if the values are higher. In the thesis it is aimed to construct a neural network that can predict images for 3D construction, point cloud. That is way these traditional metrics do not work well in this task and that is why an evaluation metric of average chamber distance (CD) change is used. The chamfer distance value of a point cloud calculated with ground truth images is compared to the point cloud calculated with all the images in the subset. This chamfer distance is taken as reference chamfer distance value. The point cloud calculated from predicted images from the neural network is compared to the point cloud calculated with all the images in the subset. Two chamfer distances are then compared to each other. If chamfer distance with predicted images is smaller than reference chamfer distance, the predicted images are then better for point cloud construction than reference images. Chamfer distance change in Figure 16 is zero if predicted images are equally good for point cloud construction that ground truth images, if value is negative predicted image produce better point cloud that ground truth images. The observation that some of the predicted results are slightly better than the given ground truth, indicates that the neural network performance could benefit from more suitable training data labeling. Still, even with the current labeling the trained model is capable of performing the intended task. From all the subsets, the average change in chamfer distance is taken. Based on chamfer distance change low number of hidden units predict as well as higher amount of hidden units. But also, this metric does not describe well enough the results, because lower amount of hidden units only learns the middle images to be zero in the image subset. They only learn the beginning of the subset and the end. When the hidden unit amount is 500 and 1000 the networks learn more about the middle images of the subsets, but they do not predict them correctly. When hidden unit amount is increased also the middle images in the subsets are predicted better.

6.2 Chamfer distance threshold examination

By changing the chamfer distance threshold in training data, the label balance can be changed. Figure 17 shows the label balance with two different chamfer distance thresholds. On the left-hand side, there is the class label balance of the training and the test data that has been constructed with chamfer distance threshold of +5 %, meaning that the chamfer distance can increased compared to the previous added image and still it is determined as an image that should be saved. On the right hand side of the Figure 17, there is a class balance of the training data that has a chamfer distance threshold of -25 %, so the chamfer distance has to decrease 25 % compared to a previous saved image before it is determined to be saved image.

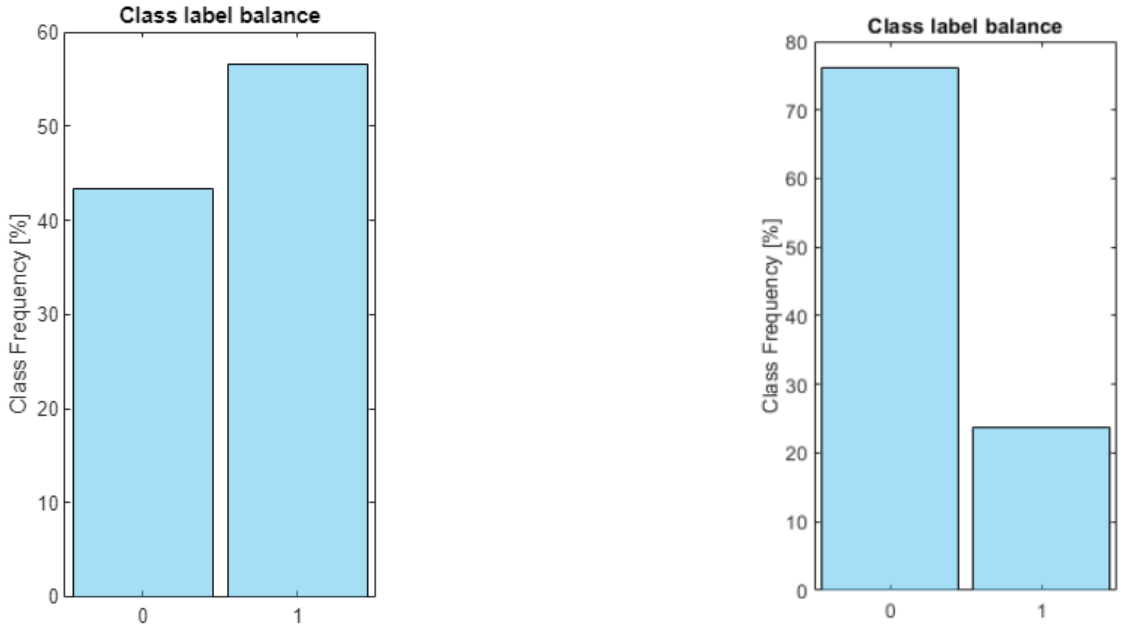

Figure 17. The balance of class labels in training data with chamfer distance threshold of +5 % on the right side of the figure and -25 % on the left side.

Using different training and testing data to the same neural network structure, the 1 layer LSTM network structure with 2000 hidden units from Chapter 6.1 is used, the effect of

chamfer distance threshold value to evaluation metrics, can be seen as shown in Figure 18. The best chamfer distance change is achieved with data that is constructed with the chamfer distance threshold of -7.5 %. With traditional metrics the best threshold value would be different when they only describe how well the prediction compares to ground truth values.

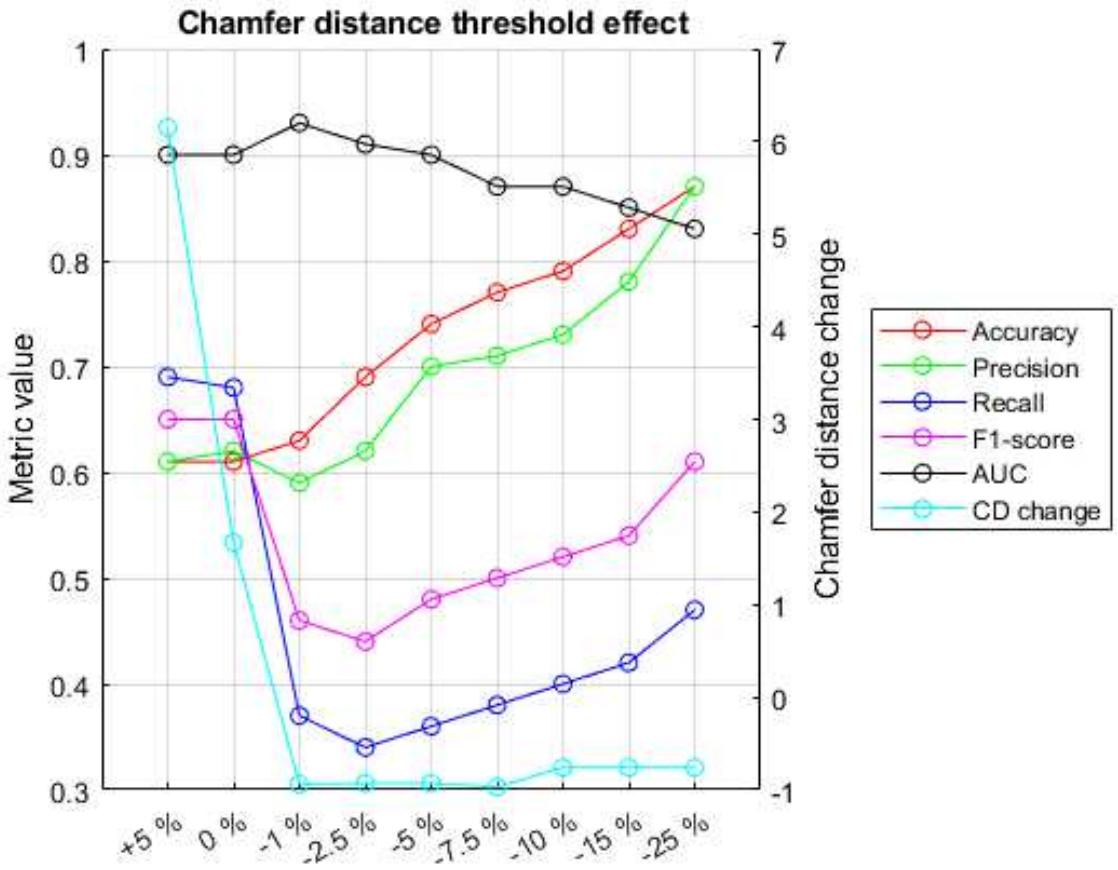

Figure 18. Chamfer distance threshold difference effect to predictions.

By taking one image subset of the test data to examination, can be seen how network prediction picks images to be saved. One image subset is shown in Figure 19 with a test data for that image subset and predicted values of that image subset. On left hand side y-axis 0 means that image is not saved and 1 means that it is saved. The chamfer distance values after each added image can also be seen in Figure 19 on yellow in right hand side y-axis. In this case, the networks fail to pick some of the images that reduce the chamfer

distance considerably, like image number 21. On the other hand, the network does not pick any of the images that would increase the chamfer distance considerably.

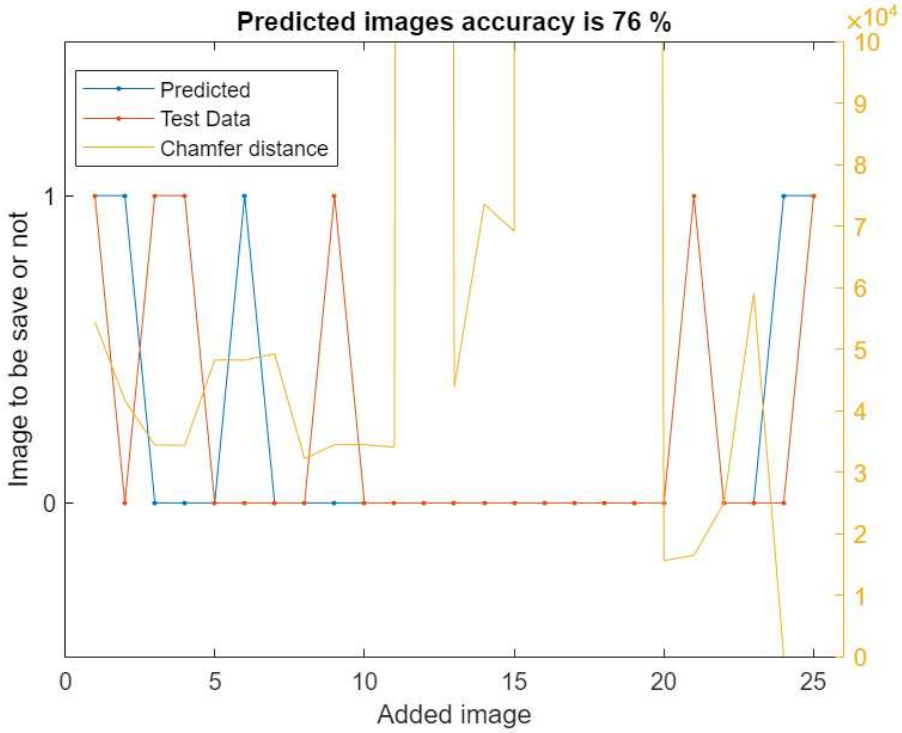

Figure 19. Predicted images compared to ground truth images and the chamfer distance change.

6.3 Chamfer distance loss function

As shown in Figure 14, a minimal chamfer distance is achieved with different set of images. It does not need to be the same exact images that needs to be recognised to minimize chamfer distance. To be able to pick one, but only one, of the images 13, 14 and 15, for example, an own loss function is investigated, which would minimize the chamfer distance of the predicted image. The chamfer distance loss function is used to compare the point clouds to each other. In Figure 20, it is shown a loss function idea. Predicted labels after each iteration are received into loss function together with the ground truth labels. From those predicted labels, images are picked based on whether an image is predicted

to be a valuable for the point cloud construction or not. The point clouds are calculated with SfM with all the predicted images. The point clouds are compared to a point cloud calculated with all the images in each subset of images. The loss of the iteration is based on average chamfer distances of all the training subsets.

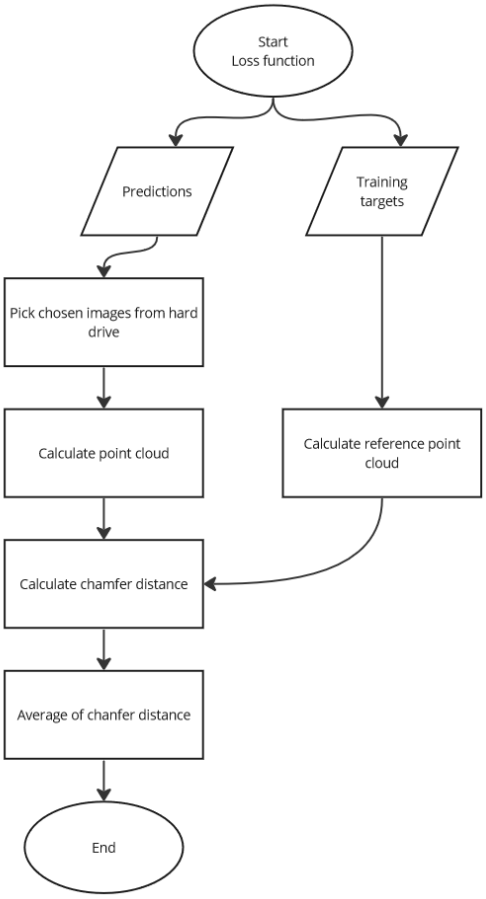

Figure 20. Plan for loss function to minimize chamfer distance of predicted images.

An issue with the loss function, described in Figure 20, is that in the backpropagation of the chamfer distance loss function is not a partial derivative respect to predictions as in Equation 19. Other issue is shown in Table 2. On the top row of the table is, an image number. In this example, there is only ten images in the image subset. A reference point cloud is calculated with all the images, shown on the second row. On the third row, are shown the images that the network has predicted for one iteration. To calculate

a point cloud at least two images are needed, which means that the first point cloud can be calculated after image 6, shown in row 4. To calculate chamfer distance change two chamfer distance values are needed, the first chamfer distance change value can be calculated to image number 9, shown in row 5. And also, backpropagation can only be signed to images 9 and 10, shown in row 6. In the backpropagation, a correction value should be signed to each sample, which is used to correct weights. Images are fed into the network one by one. That is why the chamfer distance should be calculated after each added image to get the chamfer distance change after each image and, based on the chamfer distance changes, the back propagation change value would be calculated. The problem is that the chamfer distance can be calculated after at least two images have been predicted, so what should be the backpropagation correction if chamfer distance cannot be calculated for images 1 to 8. In this work, an assumption was used that the first image should always be selected. Because of that, the correction can be signed to the first image. The correction value for the images between images 2 to 8 is hard to define with only the chamfer distance value.

Table 2. How to deal with the backpropagation factors for each image.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
All images	x	x	x	x	x	x	x	x	x	x
Predicted images			x			x			x	x
Calculated point cloud based on predicted images	-	-	-	-	-	1	-	-	2	3
Calculated chamfer distance change	-	-	-	-	-	-	-	-	1	2
Backprobagation correction based on chamfer distance chance	-	-	-	-	-	-	-	-	1	2

As a solution for these issues, a loss function shown in Figure 21 is proposed. In that, the

chamfer distance is used to check whether a concerned image would be selected or not. Then the training ground truths are updated to correspond iteration predictions. That way the earlier images can be also checked that they are correctly predicted and issue with predicting one of Figure 14's images 13, 14 and 15 would be equally as good as long as only one of those images is included in predicted images. After the ground truths are updated, cross-entropy loss is used for the predicted images and updated ground truths. With this solution, backpropagation correction issue can also be handled.

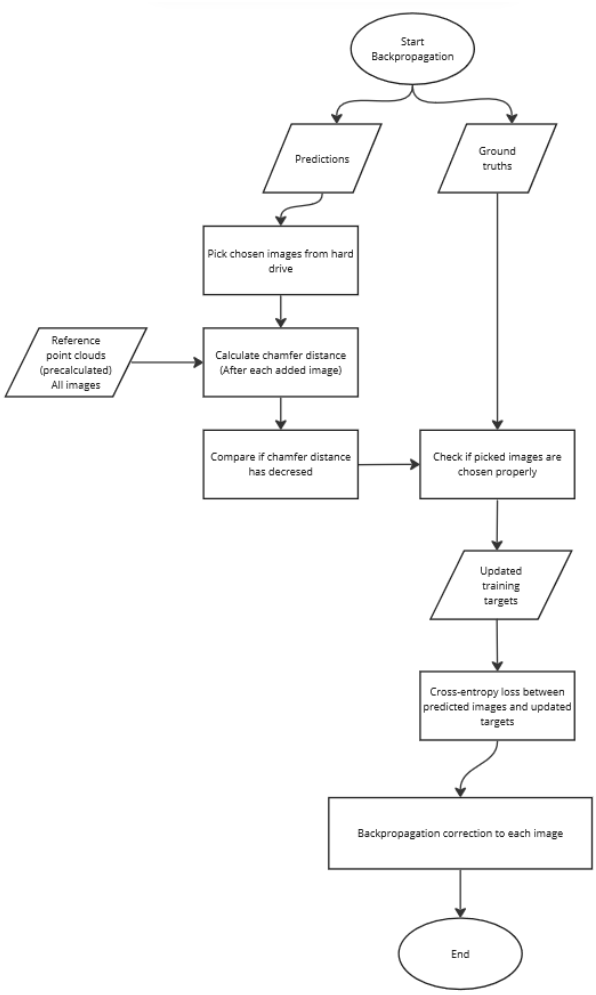

Figure 21. Used loss function to minimize the chamfer distance of the predicted images.

With own loss function, the network performance is worse than with the binary cross-

entropy loss. The chamfer distance change is 0.08. With the test data accuracy being 0.76, precision 0.81, recall 0.31, F1-Score 0.45 and AUC 0.84. The chamfer distance loss function fails to predict images to be saved in the middle of the image subset as seen in Figure 22.

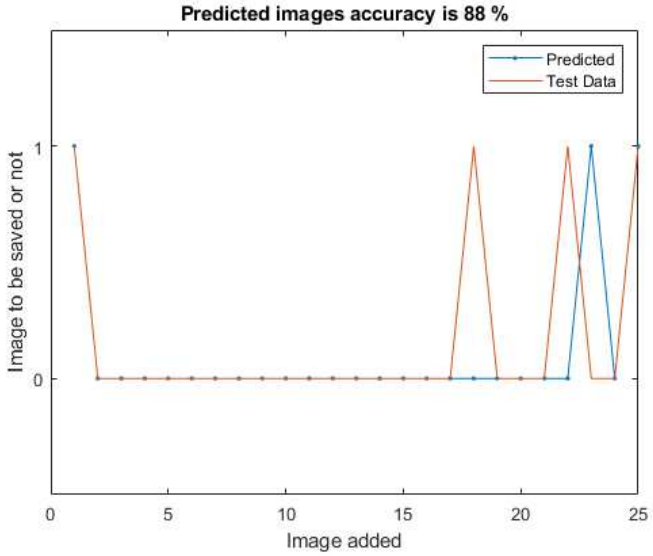

Figure 22. The predicted images compared to the ground truth images.

Loss function with point cloud calculation from predicted images is time consuming to calculate which makes the neural network training to take longer than with cross-entropy loss. As ground truths are changing between epochs, the loss is fluctuating a lot during the training.

6.4 One image prediction from image set

The training data is also constructed to try out if the images could be selected from an image batch using cross-entropy loss. An image subset of ten images is created and the chamfer distance after each image is used to determine which image gives the biggest decrease in the chamfer distance in comparison to the previous image and that image

is labelled as an image to be saved. The first image is not selected because, to calculate the point cloud, at least two images are needed. The last image is also omitted from the selection. Figure 23 shows how labels between different images are distributed. Indexing is so that second image in the image subset is indexed as 1 and ninth image is indexed as 8. On the left side, labels are without a sampling and on right hand side are labels with an oversampling to avoid imbalanced labels.

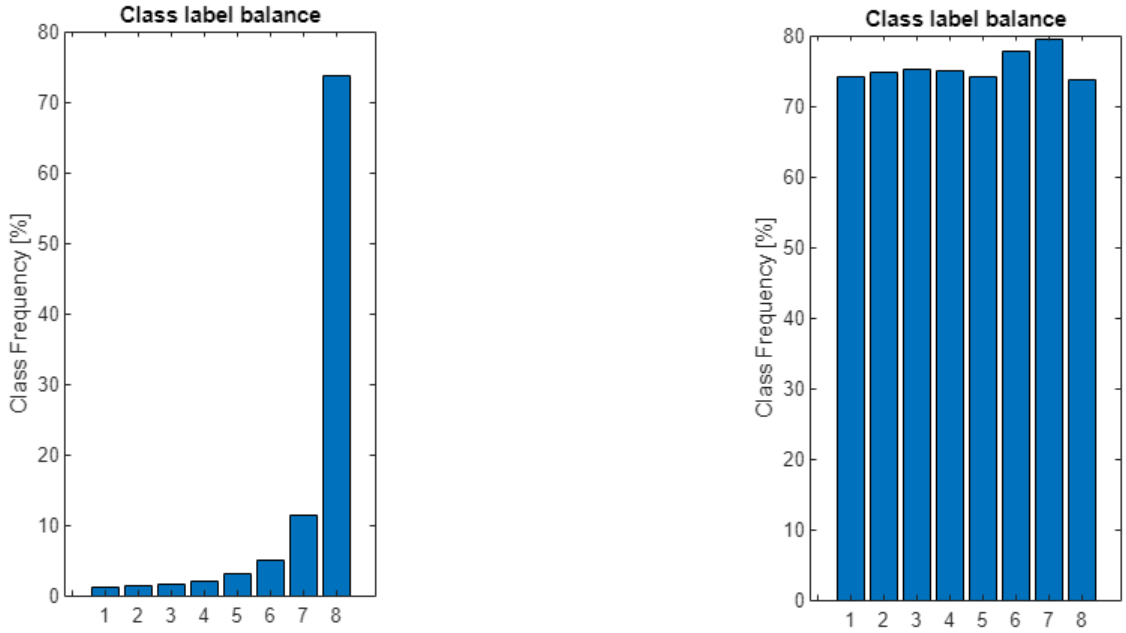

Figure 23. The balance of class labels of 10 image set of images on the left side and labels with oversampling on the right side.

The best result achieved with the test accuracy was 0.62. Training and validation accuracy was 0.94 so the model was overfitting even though training data was oversampled and L2 regularization was used to mitigate overfitting. From the confusion matrix in Figure 24 can be seen that the model has learned always to predict one of the last images.

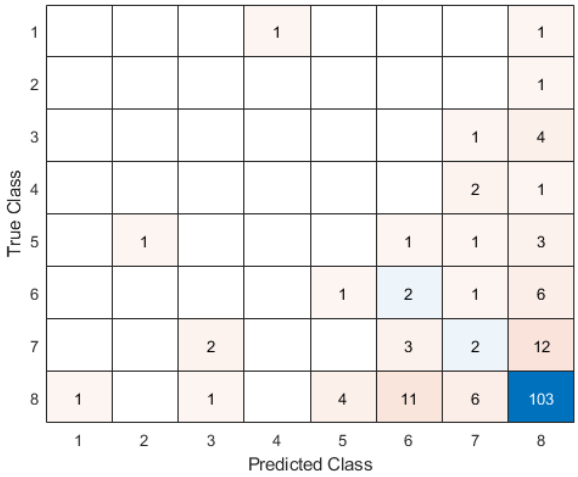

Figure 24. Confusion matrix of the prediction results.

To get a neural network to learn more of the general structure, more training data was investigated. Now the training data had same image subsets several times with a different labelling. Labels were selected so that, for example, images 13, 14 and 15 in Figure 14 would be selected in the different training data sequences. In one sequence, image 13 would be selected and images 14 and 15 not. In the other training data sequences, the same predictors are used but this time image 14 would be selected and images 13 and 15 not and so on. By adding more training data, using same predictors with different labels, did not affect the model predictions at all as seen in Table 3

Table 3. More general training data effect to model predictions.

	Data with same predictors with different ground truths	**Data with one ground truth to predictors**
Accuracy	0.77	0.77
Precision	0.71	0.70
Recall	0.38	0.38
F1-score	0.5	0.49
AUC	0.87	0.87
Chamfer distance change	-0.97	-0.97

7 Results

In this thesis, it was investigated how the neural networks can select an image with important information for the point cloud generation without losing point cloud accuracy. The chamfer distance is used as a measure to label training data, which is built with vSLAM and SfM. LSTM neural networks were used to predict informative images from the image subset. Although the chamfer distance between point clouds constructed from predicted images and all the images in the subset, were smaller, in some cases than the chamfer distance between the point clouds constructed from ground truth images and all the images in the subset, models lack accuracy. The models were not able to learn from the predictors generally the chamfer distance changes. When the neural network has more of a hidden units in several layers it will learn more precisely the chamfer distance changes as shown in Figure 25. In the Figure 25 on x-axis is images added. The predicted labels for the images are shown in blue and test data labels in orange. On y-axis 0 means that image is not saved and 1 means that it is saved. The model is also able to predict images in the middle of the subset.

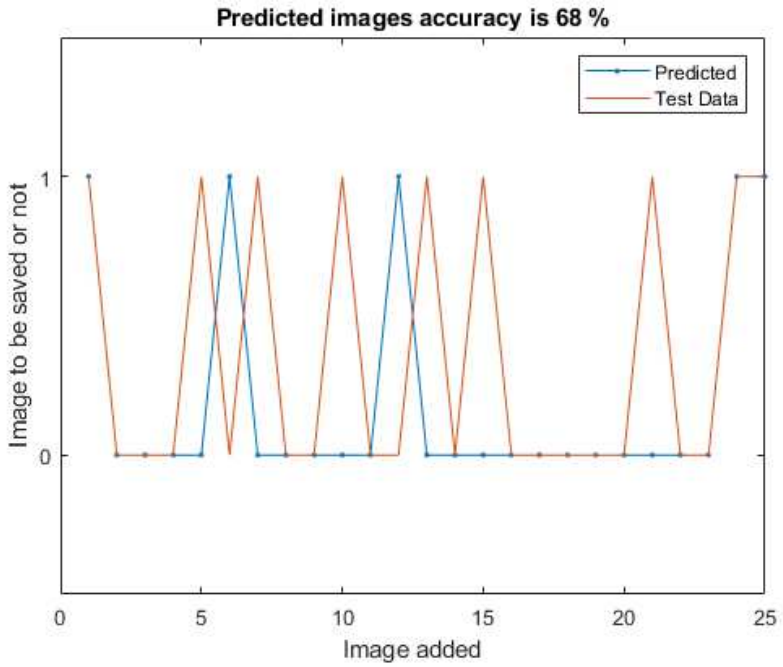

Figure 25. Predicted images compared to ground truth images with higher amount of hidden unit.

With a more simple network only image subset beginning and ending can be predicted correctly, Figure 26. In the middle of the subset, all the images are only predicted not to be important to be saved.

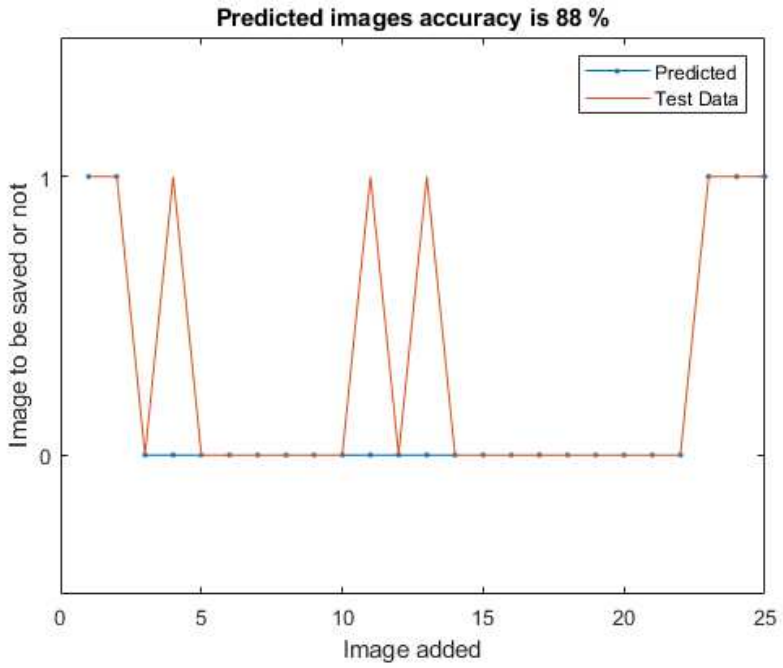

Figure 26. Predicted images compared to ground truth images with lower amount of hidden units.

From the results (Figure 16 and Figure 18) can be seen that in some case chamfer distance is much smaller with predicted images than with ground truth images. As an average neural network chose less images to be saved than there were ground truth images, still, in many cases chamfer distance between point clouds from predicted images and from all the images was smaller than with ground truth images. Based on this simple neural network used in the thesis was able to choose better images than ground truth images for 3D reconstruction. But this is probably more due to that chamfer distance change alone is not enough to determine if image has important information for point cloud calculation or not, but it needs also to have some additional information combined with it to be able to determine more precisely if image has important information for point cloud calculation or not. As an example, in Figure 27 is shown one case where there were six ground truth images and the neural network predicted six images. They are not the same images. The chamfer distance from the point cloud calculated from the predicted images is about 50 % smaller than the chamfer distance from the point cloud calculated from the ground truth images when comparing to the point cloud calculated with all the

images in the subset.

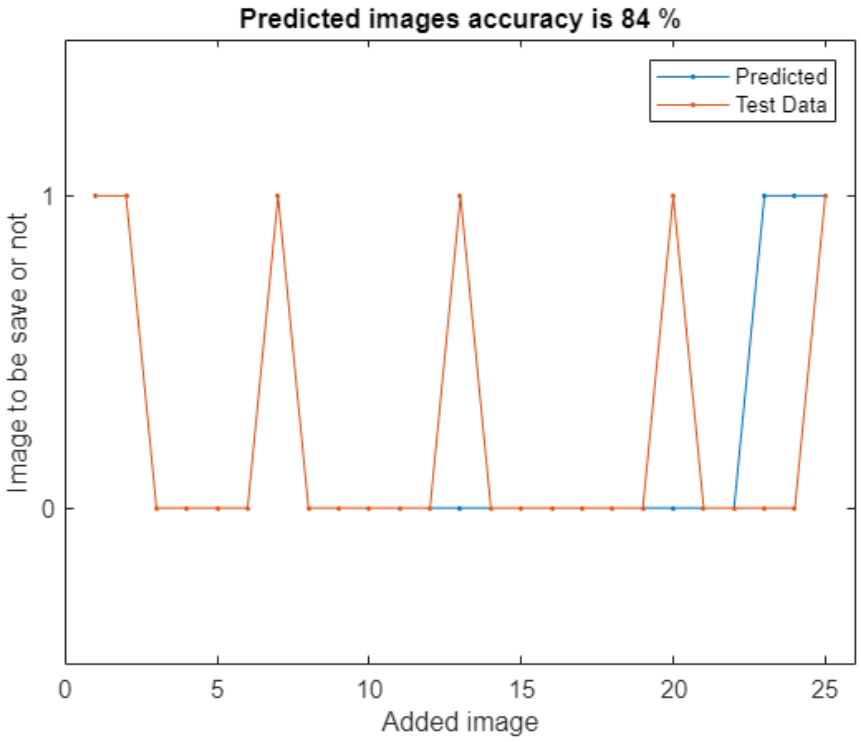

Figure 27. Predicted images compared to ground truth images with smaller chamfer distance value.

With the best model found in this thesis the point cloud computing time can be reduced by 68 % when comparing to the point cloud calculation with all images without losing too much of the accuracy of the point cloud, when the image sequence is relative short. Direct comparison to previous investigations is hard to find. Chen et al. (2020) presented that choosing important frames accurately in real-time on mobile platforms with the improved CNN model called PWC-Net is possible, but they used separate neural network structure to prevalidate the input images before using PWC-Net. Feng et al. (2019) presented the 2D3D-MatchNet, which is end to end a triplet-like deep neural network architecture that learns the descriptors from 2D images and key points from 3D point clouds for visual pose estimation. 2D3D-MatchNet localization results outperform ORB-SLAM2. DeepI2P is deep neural network used to cross-modality registration of 2D image and 3D

point cloud developed by Li and Lee (2021). It classifies whether the projection of each point in the point is in front of the camera or behind it. Labelled points are then directed into an inverse camera projection solver to estimate the relative pose (Li & Lee, 2021). Based on these studies and this thesis the neural network can be used to predict images that are important to point cloud reconstructing, but better results could be achieved by further development of the training procedure.

8 Conclusion and future work

In the thesis, it was investigated how the neural networks recognise images with important information for the point cloud generation without losing the point cloud accuracy. The chamfer distance is used as a measure to label training data, which is built with vSLAM and SfM. LSTM neural network were able to predict key images from the image subset with more complex networks with a higher precision. With the best model found in this thesis, the point cloud calculating time can be reduced by 68 % when comparing to point cloud calculation with all images without losing too much of the accuracy of the point cloud, when the image sequences are relatively short. The learning of the general concept needs to be improved. To learn more about generally chamfer distance change effect, some other concept might be needed apart from supervised learning, because there are multiple ground truths that produce equally good end results. Reinforced learning that does not use any labelled data or semi-supervised learning that used some labelled data and a lot of non labeled data could learn better the general concept of minimising the chamfer distance of image sequences. Predictor influence on the neural network prediction accuracy should also be investigated individually to see which predictors have the most correlation with the choosing of images with important information for 3D point cloud construction. Other predictors than used camera movement and match feature points between images could be considered, for example, statistical information of feature points in each image.

Bibliography

Ai, Y.-b., Yang, X.-q., He, J.-l., Fu, L., Li, J.-b., & Lu, M. (2021). Visual slam in dynamic environments based on object detection. *Defence Technology*, *17*(5), 1712-1721.

Alexiou, E., & Ebrahimi, T. (2017). On subjective and objective quality evaluation of point cloud geometry. *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*.

Alpaydin, E. (2014). *Introduction to machine learning 3rd edition*. The MIT Press, Cambridge, Massachusetts.

Barrow, H., Tenenbaum, J., Bolles, R., & Wolf, H. (1977). Parametric correspondence and chamfer matching: Two new techniques for image matching. *Proc. 5th Int. Joint Conf. Artificial Intelligence*, 659–663.

Besl, P., & McKay, N. D. (1992). A method for registration of 3-d shapes. *INternational Journal of Geo-Information*, *14*, 239-256.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+Business Media, LLC.

Buduma, N., & Lacascio, N. (2017). *Fundamentals of deep learning designing next-generation machine intelligence algorithms*. O'Reilly Media Inc.

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). BRIEF: Binary robust independent elementary features. *ECCV 2010: Computer Vision – ECCV 2010*, *11th European Conference on Computer Vision Heraklion*, *Crete*, *Greece*, *September 5-11*,(Part IV), 778-792.

Chen, Y., Yan, L., & Lin, X. (2020). An automatic key-frame selection method for visual odometry based on the improved pwc-net. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Informatio, XXIV ISPRS Congres, XLIII-B2-2020*.

Computer Vision Group, T. U. o. M., TUM Department of Informatics. (2022). *Dataset download of computer vision group, tum department of informatics, technical university of munich*. Retrieved from [2022-12-09]https://vision.in.tum.de/data/datasets/rgbd-dataset/download

da Silva Cruz, L. A., Dumić, E., Alexiou, E., Prazeres, J., Duarte, R., Pereira, M., … Ebrahimi, T. (2019). Point cloud quality evaluation: Towards a definition for test conditions. *QoMEX*.

Fan, H., Su, H., & Guibas, L. (2017). A point set generation network for 3d object reconstruction from a single image. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Feng, M., Hu, S., Ang Jr, M. H., & Lee, G. H. (2019). 2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud. *2019 International Conference on Robotics and Automation (ICRA)*.

Good, I. J. (1952). Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, *14*, 107-114.

Hoang, D.-C., Lilienthal, A. J., & Stoyanov, T. (2020). Object-rpe: Dense 3d reconstruction and pose estimation with convolutional neural networks. *Robotics and Autonomous Systems*, *133*(6).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735-1780.

Hu, S., Feng, M., Nguyen, R. M. H., & Lee, G. H. (2018). Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Häming, K., & Peters, G. (2010). The structure-from-motion reconstruction pipeline – a survey with focus on short image sequences. *Kybernetika*, *46*, 926-937.

Javaheri, A., Brites, C., Pereira, F., & Ascenso, J. (2017). Subjective and objective quality evaluation of 3d point cloud denoising algorithms. *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*.

Kendall, A., & Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kendall, A., Grimes, M., & Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small ar workspaces.

2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality.

Li, J., & Lee, G. H. (2021). Image-to-point cloud registration via deep classification. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Loomis, L. H., & Sternberg, S. (1990). *Advanced calculus*. Jones and Bartlett publishers, Boston.

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216-231.

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2021). A comparison method for 3d laser point clouds in displacement change detection for arch dams. *INternational Journal of Geo-Information*.

Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. SIAM, Philadelphia.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., & Sayd, P. (2006). Real time localization and 3d reconstruction. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, *1*, 363-370.

Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, *31*(5), 1147-1163.

Mur-Artal, R., & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, *33*(5), 1255-1262.

Nazir, D., Afzal, M. Z., Pagani, A., Liwicki, M., & Stricker, D. (2021). Contrastive learning for 3d point clouds classification and shape completion. *Sensors*.

Nielsen, M. A. (2019). *Neural networks and deep learning*. http://neuralnetworksanddeeplearning.com: Determination Press.

Ping, G., Esfahani, M. A., Jiaying, C., & Han, W. (2022). Visual enhancement of single-view 3d point cloud reconstruction. *Computers Graphics*, *102*, 112-119.

Prince, S. J. (2012). *Computer vision: models, learning and inference*. Cambridge University Press. Retrieved from http://www.computervisionmodels.com.

Qin, T., Li, P., & Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, *34*(4), 1004-1020.

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection.

ECCV 2006: Computer Vision – ECCV 2006, 9th European Conference on Computer Vision Graz, Austria, May 7-13(Part I), 430-443.

Rublee, E., Rabaud, V., Konolige, K., & Brandski, G. (2011). ORB: an efficient alternative to SIFT or SURF. *IEEE 2011 International Conference on Computer Vision*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536.

Saabni, R., & El-Sana, J. (2011). Word spotting for handwritten documents using chamfer distance and dynamic time warping. *Proceedings of SPIE - The International Society for Optical Engineering*, *7874*, 1-10.

Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM a tutorial into long short-term memory recurrent neural networks. *Cornell University*.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 573-580.

Sun, D., Yang, X., Liu, M.-Y., & Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Szeliski, R. (2021). *Computer vision: Algorithms and applications 2nd edition*. Springer. Retrieved from https://szeliski.org/Book/

Tan, W., Liu, H., Dong, Z., Zhang, G., & Bao, H. (2013). Robust monocular slam in dynamic environments. *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

The MathWorks, I. (2022). *Mathworks documentation*. Retrieved from [2022-08-31]https://se.mathworks.com

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T., & Yeung, S.-K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1588-1597.

Wang, X., Ang, M. H., & Lee, G. H. (2021). Point cloud completion by learning shape

priors. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Wen, F., Zou, D., Miao, R., Ying, R., & Liu, P. (2019). Efficient outlier removal for large scale global structure-from-motion. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yan, W. Q. (2021). *Computational methods for deep learning theoretic, practice and applications*. Springer. Retrieved from https://link.springer.com/book/10.1007/978-3-030-61081-4

Zhuang, Y., Rui, Y., Huang, T., & Mehrotra, S. (1998). Adaptive key frame extraction using unsupervised clustering. *Proceedings 1998 International Conference on Image Processing*.