ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney Research

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Match Correspondence in Vegetation Point Clouds using Deep Learning

Autumn Term 2018

Supervised by:
Tejaswi Digumarti
Dr. Cesar Cadena Lerma
Dr. Paul Beardsley

Author:
Benedikt Dietz

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Match Correspondence in Vegetation Point Clouds using Deep Learning

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Dietz	**Benedikt**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, Schweiz 23/07/2018	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Acknowledgments

This semester project was conducted at Disney Research, Zurich as a collaboration with Prof. Roland Siegwart's 'Autonomous Systems Lab' at the Swiss Federal Institute of Technology (ETH) Zürich. I would like to extend my sincerest gratitude to Prof. Siegwart and Disney Research for giving me the opportunity to participate in this research as well as to my supervisors Dr. Paul Beardsley, Dr. Cesar Cadena Lerma and Tejaswi Digumarti for their guidance and support throughout the project.

Abstract

Reconstruction of three-dimensional objects from images has been a field of great interest for some time. Despite many advancements, vegetation still poses a considerable challenge on state-of-the-art approaches and is often ignored. However, there is still a demand for improved solutions for various applications such as three-dimensional modelling for the entertainment industry as well as for agricultural purposes like robot picking, carbon sequestration measurements and other ecosystem monitoring approaches. Recently an approach utilising a siamese deep convolutional neural network has shown success with 'learnt' three-dimensional geometric feature descriptors [1]. In order to evaluate the performance considering vegetation data, a new data generation pipeline was set up to be able to acquire arbitrary amounts of training data or, optionally sample online during training. Training and validation testing at his point has been done on synthetic data for tractability ans as a step towards processing real data in future works. The network architecture used by Zeng et al. [1] was adapted, built and optimised. Results on validation data show a quickly decreasing error rate to consistent values of below $\sim 0.2\%$ *within the first few epochs. Error divergence in later training epochs is explained with the objective of the loss function on the means, rather than outliers or variance. Lastly, reconstruction is tested on a a set of frames showing generally good results depending on decision rule, however with the considerable drawback of very unevenly distributed correspondences. Estimation of the relative transformation between frames is computed using a singular value decomposition (SVD) approach combined with random sample consensus (RANSAC).*

Symbols

Symbols

x_0	Input 1: Initial random points
x_+	Input 2: Correspondence points
x_-	Input 3: Non-match points

Acronyms and Abbreviations

TDF	Truncated Distance Function
SVD	Singular Value Decomposition
RANSAC	Random Sample Consensus
ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab

List of Figures

Chapter 1

Introduction

Capturing three-dimensional objects has been a field of great interest in computer vision and graphics for a long time. The evolution of camera and range sensing technologies has spawned a number of applications including object retrieval, 3D reconstruction, pose estimation and camera localisation. In this context, the term '3D Reconstruction' references the process of capturing and reconstructing the shape and appearance of a given object from images or measurements. Due to a rising interest for three-dimensional models for applications ranging from agricultural robotics and ecosystem monitoring to high-quality models for the entertainment industry, this field of research has attracted significant interest in recent years, accompanied by frequent technical advancements.

A common approach to 3D reconstruction from images is to identify correspondences in multiple frames. In general, if a number of correspondences can be identified which is sufficient to fully determine the camera pose with six degrees of freedom (3D translation and 3D rotation), the respective transformation between the two corresponding frames can be reconstructed. Due to occurrences of outliers and defects in the correspondences, larger amounts are sampled and evaluated under considerable computational effort to estimate the transformation. There is a wide range of mostly histogram derived, hand-crafted geometrical feature descriptors that may be applied to identify correspondences [2, 3], however these tend to fail with noisy, incomplete surfaces as acquired from RGBD images [1]. Vegetation is particularly challenging for this task due to its self-similar and repetitive structure, significant amount of occlusions and uniformity of colour.

Recently, Zeng et al. [1] proposed a method to learning three-dimensional feature descriptors utilising a deep learning approach, called '3DMatch'. To this end, a Siamese convolutional network is designed where each branch takes in the volumetric region around an arbitrary point of interest and computes an embedding (or descriptor). The euclidean distances of two descriptors are interpreted as a measure of similarity and used to identify correspondences. The approach has been successfully trained and tested using indoor scene images. Hence the motivation for this work was to evaluate and optimise the performance of the 3DMatch approach when employed on RGBD image sequences of trees. The general idea is to equip a drone with a RGBD sensor and have it flying around a tree, generating image sequences which will later be used to both train the neural network as well as to reconstruct the respective tree using the acquired correspondences. Due to a lack of data from real trees at this point in time, images of synthetic trees (SpeedTree [4])were used to evaluate the performance and potential and lay the foundation for future work.

Chapter 2

Theory

2.1 Related Work

As previously mentioned, reconstruction through correspondence matching has been a field of interest for computer vision and graphics applications. The related work concerning both hand-crafted and learned descriptors for three-dimensional objects is briefly reviewed in the following section.

Hand-crafted local feature descriptors have made considerable progress and been successful in identification of correspondences for smooth and complete surfaces. These are based on a range of approaches from Spin Images using surface meshes [5], Signatures of Histograms [6] or Geometry- and Feature Histograms [7, 8]. However they struggle with incomplete and noisy data as acquired from current commodity range sensors. Additionally, due to their target data specific design, they generalise poorly and cannot be used on different data and object settings without considerable effort invested in calibration which motivates the used of learnt feature descriptors as presented in this work.

Concerning learnt three-dimensional descriptors, there has been much progress with learning geometric representations for three-dimensional objects. Deep learning for modelling three-dimensional shapes has been introduced by Wu et al. [9] and several other approaches attempted deep feature extraction for the purposes of classification and object retrieval [10, 11]. However these works focus on global descriptors and representations as opposed to the local feature descriptors being learnt in [1] as well as this work. Also related, despite only operating on complete, synthetic surfaces, is the publication of Guo et al. [12] which utilises a two-dimensional convolutional network embedding to match local geometric features for mesh labelling. Contrary to this work, 3DMatch [1] manages to preserve the spatial correlation of the volumetric patches, thus leveraging the convolutional network and operates on real RGBD data.

2.2 Volumetric Descriptors

For the task of matching corresponding points on three-dimensional objects, it is essential to encode information about an arbitrary point on the object in a way that enables and optimises subsequent processing. Following the methods of Zeng et al. [1], 'Truncated Distance Functions' (TDF) were used for this purpose. A volumetric patch around the chosen point is defined and divided into voxels. The value of the TDF of a given voxel represents the distance between the voxel's centre and the

closest point on a three-dimensional surface. The resulting values are truncated, normalised and then flipped. In this representation each value is between zero and one where the former indicates a point far from any surface and the latter represents a surface point. Another common distance function is the 'Truncated Signed Distance Function' (TSDF) [13] which, in addition to surface and off-surface points also encodes occlusions. However TDFs were chosen since sacrificing the distinction between free space and occlusions provides an essential advantage by ensuring that the largest gradients between TDF voxel values are always concentrated around the surface which is crucial to the robustness of the descriptor [1].

2.3 Neural Networks and Deep Learning

Neural network or 'Deep Learning' approaches have gained much interest inside as well as outside of the scientific community. In the following section, the basic concepts involved in the presented work are briefly outlined.

2.3.1 Neural Networks

Arguably the history and successes of 'Artificial Neural Networks' (ANN) began with the introduction of the 'Perceptron' by Rosenblatt [14] in the late 50s of the last century, who successfully built an ANN inspired by- and modelled after a simplified model of biological neurons in real brains. The neurons process incoming data and pass it to the subsequent layers after a non-linear 'activation function'. Through an iterative and often 'supervised' process, the randomly initialised parameters of the 'hidden' nodes between input and output are optimised, i.e. 'learnt'. This idea has come a long way since its early days with many advancements and great scientific interest, particularly in recent years.

Deep Learning

'Deep Learning' refers to the same idea as neural networks, however multiple layers of hidden nodes are used in order to be able to learn a range of representations, corresponding to different levels of abstraction. This approach has had much success in a wide range of applications ranging from face- or speech- recognition [15, 16] and natural language processing [17] to computer vision [18] and others.

2.3.2 Convolutional Networks

Convolutional networks are a sub-category of neural networks in general. The name refers to the main operation of the layers, namely convolutions. While for conventional, fully connected layers, the forward propagation of a particular layer is given by

$$F(x) = \phi(Wx + b) \tag{2.1}$$

where $F(x)$ is the output of the considered layer, W and b represent the weights and bias and ϕ specifies the respective activation function. x is the input to the layer, respectively the output of the previous layer. The discrete convolution in the individual layers of a convolutional network are defined as

$$(f * h)[u] = \sum_{t=-\infty}^{\infty} f[t]h[u-t] \tag{2.2}$$

For the example of 2D or 3D image recognition, f represents the image or input data and h is the (fast decaying) kernel. For three-dimensional data as considered in this project, the kernel is a three-dimensional tensor of specified size. This tensor is moved along the dimensions of the input and computes the respective transformation for the currently considered area, thus preserving the spatial correlations. This transformation is fed to the activation function of the layer. Thus a tune-able number of 'filters' is computed in each layer of the network. For the sake of robustness to invariances, pooling layers are generally integrated. The equivalence of the weights in a general fully connected layer are the entries of the kernels in the individual convolution layers.

2.3.3 Siamese Neural Networks

Siamese network architectures have first been introduced at the beginning of the 1990s for fingerprint- and signature verification by Baldi and Chauvin [19] and Bromley et al. [20]. The Intuition behind the approach is that two samples are fed to the system, each one being processed by an individual sub-network with shared, i.e. synchronised weights. Hence the name 'Siamese'. The two sub-networks are joined at their outputs which each represent a descriptor for the given sample. A measure of similarity can now be drawn from the euclidean distance of the sample descriptors in the embedding space and be used to identify correspondences. The system is optimised using a 'Contrastive Loss' which penalises the L_2 distance between correspondences as well as the L_2 distances of non-matches below a given threshold margin. Inspired by this architecture, the 'Triplet Network' with respective loss function has been introduced by Hoffer and Ailon [21]. Instead of two samples and a label, this approach uses three samples and respective Siamese sub-networks. The samples represent an arbitrary first one and a respective match and non-match. For most of the work published so far using the Siamese or similar approaches, the sub-networks were convolutional neural networks as originally inspired by the 'LeNet-5' network for hand-written digit recognition (LeCun et al. [22]) and numerous subsequent successes and advancements with convolutional networks.

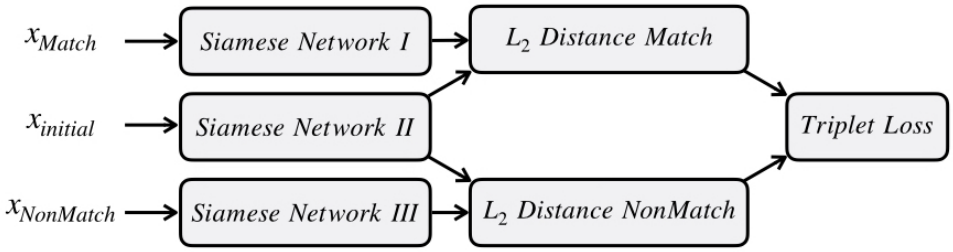

Figure 2.1: ***Graph structure for siamese network.*** *Siamese branches share weights and architecture and individually compute embedding for their corresponding input.*

2.4 Loss Function

As previously stated, the loss function is either a 'Contrastive' or 'Triplet' loss, depending on the input structure. Both take in the euclidean distance of the respective feature descriptors and are a sum of two individual losses for matches and non-matches. The first penalty is on the mean of match descriptor distances while

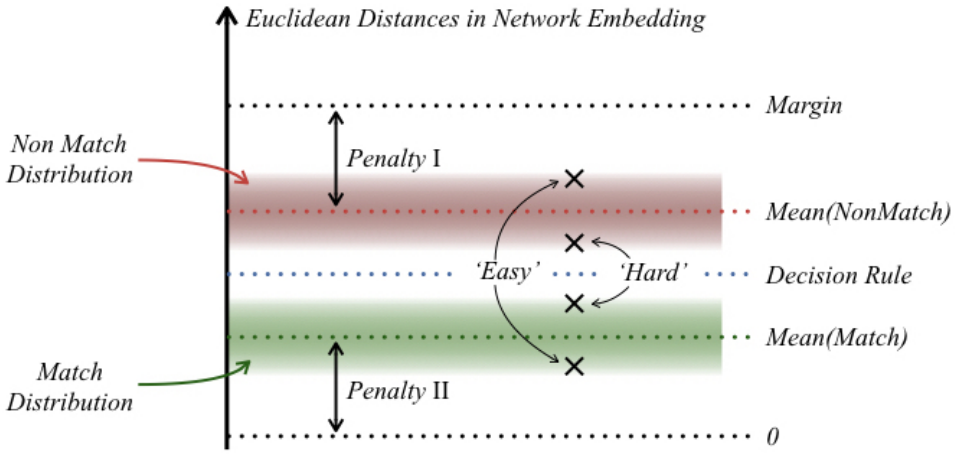

Figure 2.2: ***Intuition behind the contrastive loss.*** *The respective means of the distribution are optimised and thus pushed apart. A specified margin is used to pull non-matches away from correspondences. Pairwise euclidean distances provide the option of sampling and processing strategies. Respective 'easy' and 'hard' samples are shown in the figure.*

the second penalises the delta between non-match descriptor distances and a given threshold margin. The contrastive loss is given as

$$L_{Contrastive} = (Y - 1)\ D_F^2 + Y\ \{\max(m_{thresh} - D_F,\ 0)\}^2 \tag{2.3}$$

where Y is the label of a given pair ($Y = 0$ for matches), m_{thresh} is the threshold used to penalise the non-matches during training and D_F represent the averaged euclidean distance of a given pair of descriptors. Similarly, the triplet loss is given by

$$L_{Triplet} = D_{Matches}^2 + \{\max(m_{thresh} - D_{NonMatches},\ 0)\}^2 \tag{2.4}$$

Figure 2.2 tries to give some intuition for the aforementioned loss function. The means of the two respective distance distributions (depicted in red and green) are being pushed apart from each other during training. For the matches, the overall distance is minimised while for the non-matches, the delta between mean and a specified margin is optimised. Hence, for each processed pair of inputs, a resulting euclidean distance is computed which can be used to control the 'difficulty' of the samples used for back propagation and optimisation. One approach to this is to introduce a constant bias towards harder samples while it is also possible to dynamically control the difficulty and therefore 'steer' the learning process from more general examples towards harder and more extreme ones. Additionally, Simo-Serra et al. [23] discuss stochastic mining and sampling strategies with biases towards hard examples which proved successful compared to state of the art performances. Evaluated sampling strategies and loss variations are presented in subsequent chapters.

Chapter 3

Methods

3.1 Data Acquisition

The network is trained using truncated distance functions (TDFs) of random volumetric patches and their respective correspondences and non-correspondences. The starting point is a series of RGBD image sequences and camera poses, generated with a virtual drone flying specified trajectories around a number of randomly generated instances of synthetic Cherry trees. Synthetic trees were acquired, i.e. simulated with 'SpeedTree' [4] and the simulation was achieved using 'Unreal Engine' in addition to the 'Airsim' plugin. The virtual drone is depicted in figure 3.1. In addition to the RGDB image, a segmented version of the image is available, preliminary to distinguish the tree from the background (shape of images: (424×512)). Each frame represents a different drone position and orientation and at least two different frames are sampled for each training triplet.

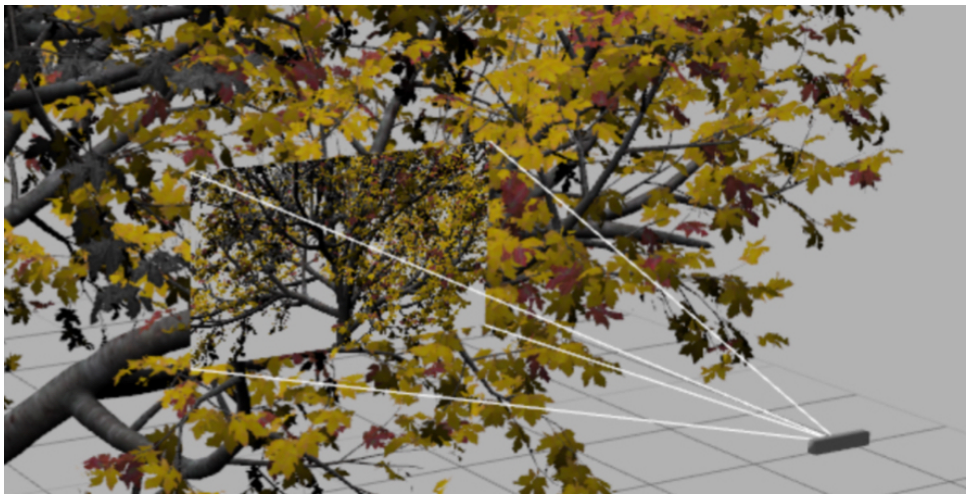

Figure 3.1: ***Simulated drone equipped with RGBD camera*** [24]

The first step in the data generation pipeline for one instance of a training triplet is to randomly sample an arbitrary tree and respective frame. Using the segmentation information, a random point on the tree in the frame is chosen. This is going to be the initial point of the triplet. Since the drone pose is known, the exact position of the initial point in the 'tree reference system' is computed and subsequently a

volumetric patch of pre-specified dimensions is extracted around the point of interest. Using the point-coordinates in the respective reference systems of the drone and the tree, a point cloud is generated and using an in-memory nearest neighbours approach, the truncated distance function values for the given volume are computed.

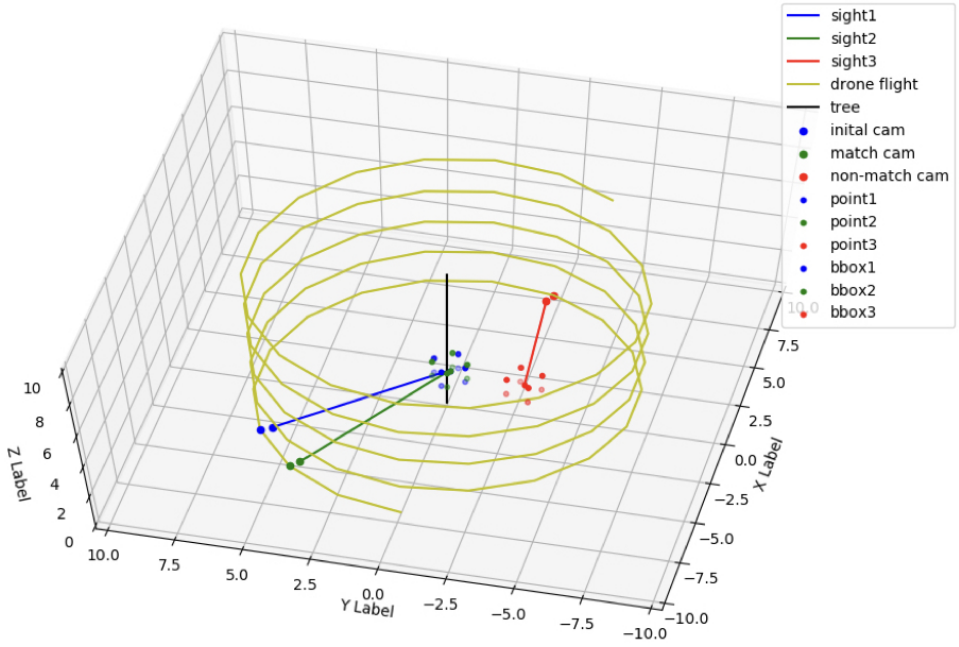

Figure 3.2: ***Drone trajectory and sampled points.*** *The trajectory of the simulated drone is plotted in yellow with the position of the tree indicated in the middle. The blue, green and yellow vectors represent the path from the respective camera position to the considered points. As shown, the first two points align and are seen from different positions while the third one (the non-match) is on an entirely different position of the tree. The resulting volumetric bounding boxes are indicated in the respective colours.*

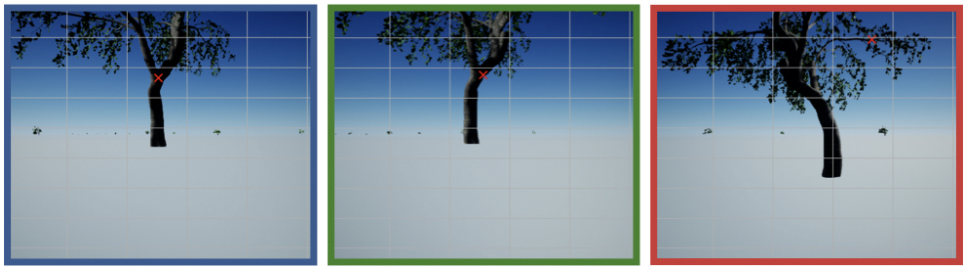

Figure 3.3: ***Resulting triplet of frames.*** *The red cross in each of the frames represents the chosen points for the triplet. The only constraints for the choice of the third point are a minimum distance of 50cm to the initial point and that the same tree is sampled as for the initial points.*

A second drone position, i.e frame is chosen randomly but with a small maximum distance constraint to the initial drone position. Using the known relative translations and rotations between individual frames, the position of the initial point in the frame and reference system of the second drone position is determined while the third point is again randomly sampled on the same tree as the initial point with

a minimum distance between the points. The conversion to TDF values follows the same principle as described for the initial point. Various checkpoints throughout the data pipeline ensure proper alignment of the corresponding points in both frames, i.e. check that the chosen point on the tree in the first frame actually aligns with the computed point in the second frame, and suppress defects due to camera frustum borders and particularly occlusions, which are especially frequent in the leaves of the trees. Figures 3.2 and 3.4 visualise the process. For the desired goal of utilising a drone capture system for real trees, this approach will have to be adapted as neither global position measurements nor relative drone transformations would be available. However, the synthetic data set used provided this information and thus ground-truth labels for given point combinations could be computed and used for the training process.

Algorithm 1 Simplified Pre-Processing Pseudo Code

trees ← list of available synthetic trees
tree.frames ← list of RGBD frames for resp. tree
tree.poses ← camera poses (global reference system)
$match\_found = 0$
$non\_match\_found = 0$
while $match\_found = 0$ **do**
    tree0 ← *trees[random(0, len(trees)]*
    frame0 ← *tree0.frames[random(0, len(tree0.frames)]*
    pose0 ← *tree0.poses[frame0]*
    depthImg0 ← load depth image *tree0.frame0*
    depthImg0 ← *depthImg0[depthImg0 > depthLimit] = 0* (i.e. invalid)
    while *cam_dist > cam_dist_threshold* **or** $frame0 == frame1$ **do**
        frame1 ← *tree0.frames[random(0, len(tree0.frames)]*
        pose1 ← *tree0.poses[frame1]*
        cam_dist ← *euclidean distance(pose0, pose1)*
    depthImg1 ← load depth image *tree0.frame1*
    depthImg1 ← *depthImg1[depthImg1 > depthLimit] = 0*
    // get random point in frame0 in cam0 system [u0, v0, w0]:
    u0, v0 ← *nonzero(depthImg1)[random(0,len(nonzero(depthImg1))]*
    w0 ← *depthImg0[u0,v0]*
    // compute [u1,v1,w1] from relative cam transformation
    if $u1, v1 > depthImg1.shape$ **then**
        break // check if point in cam frustum
    if $abs(depthImg1[u1, v1] - w1) > 0.01$ $[m]$ **then**
        break // rule out occlusions
frame2 ← *tree0.frames[random(0, len(tree0.frames)]*
pose2 ← *tree0.poses[frame2]*
depthImg2 ← load depth image *tree0.frame2*
depthImg2 ← *depthImg2[depthImg2 > depthLimit] = 0*
u2, v2 ← *nonzero(depthImg2)[random(0,len(nonzero(depthImg2))]*
w2 ← *depthImg2[u2,v2]*
if $euclidean\_distance(p0, p2) < non\_match\_tresh$ **then**
    break // enforce min. distance
p0, p1, p2 ← *point coordinates*
return *TDFgenerator(p0, p1, p2)*

In summary, this represents a pipeline for training data generation, capable of producing virtually arbitrary amounts of training instances since each individual

frame offers on-tree-points in the order of $\sim 10^5 - 10^6$ which results in an almost infinite amount of possible triplet combinations given a sufficient amount of RGBD frames. Hyper parameters to be determined and optimised for the given application are the overall dimensions of the extracted volumetric patch as well as its resolution, i.e. the size of an individual voxel. For the work presented, a $30cm^3$ patch was chosen with a voxel size of $1cm^3$. In order to considerably reduce computation times, we refrained from sampling the training instances online with each iteration but to load and save a sufficient amount of data in advance and iterate through the epochs during training. For this purpose 16,000 TDF triplets were sampled, all from the same distribution, i.e. same settings. 14,000 TDF triplets were used for training while the remaining 2,000 triplets were kept for validation. The training samples are always equally divided considering matches and non-matches as well as leaf and non-leaf points and result in two pairwise distances per triplet.

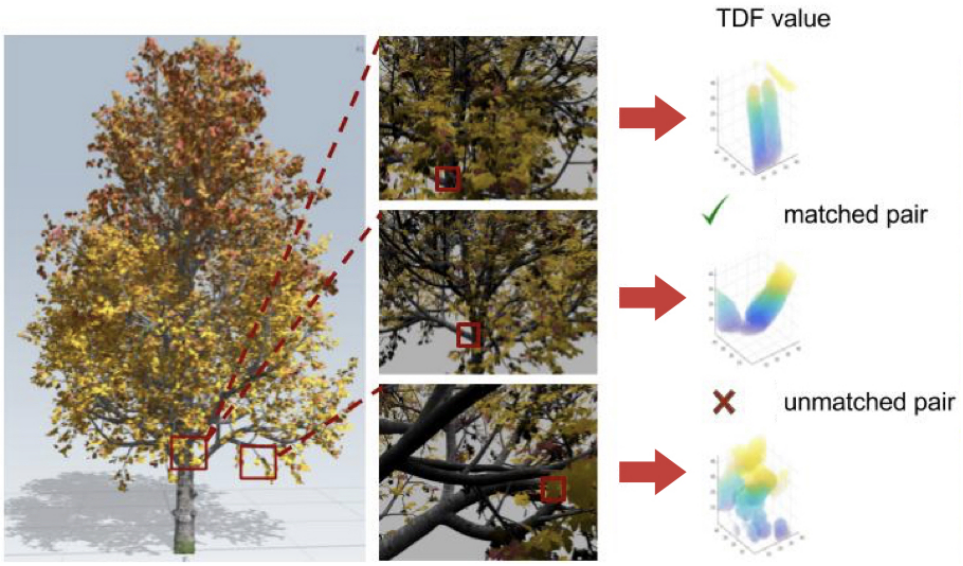

Figure 3.4: ***From RGBD image of a tree to TDF training instances.*** [24]

3.2 Framework & Training

The framework was built using *Python* and the GPU version of *Tensorflow*. Training and validation were computed with an 11GB GPU using a *CUDA* framework. Training was performed on 14,000 TDF triplets over a varying number of epochs and validation performance was evaluated on 2,000 matches and non-matches, respectively. Batch size throughout the work was set to 32. The considerable memory requirements of the three-dimensional input and extensive network architecture were a limiting factor concerning the batch sizes.

3.3 Network Architecture

The input to the network consisted of batch size $\times$ 3 TDFs, each of shape ($30 \times 30 \times 30$). The three inputs, namely initial random points (x_0), matches (x_+) and non-matches (x_-) are fed to their respective branch of the siamese network. The individual siamese branches are equal as they share architecture and weights. Figure 3.5 shows the simplified architecture of the overall network (right-hand side) and the

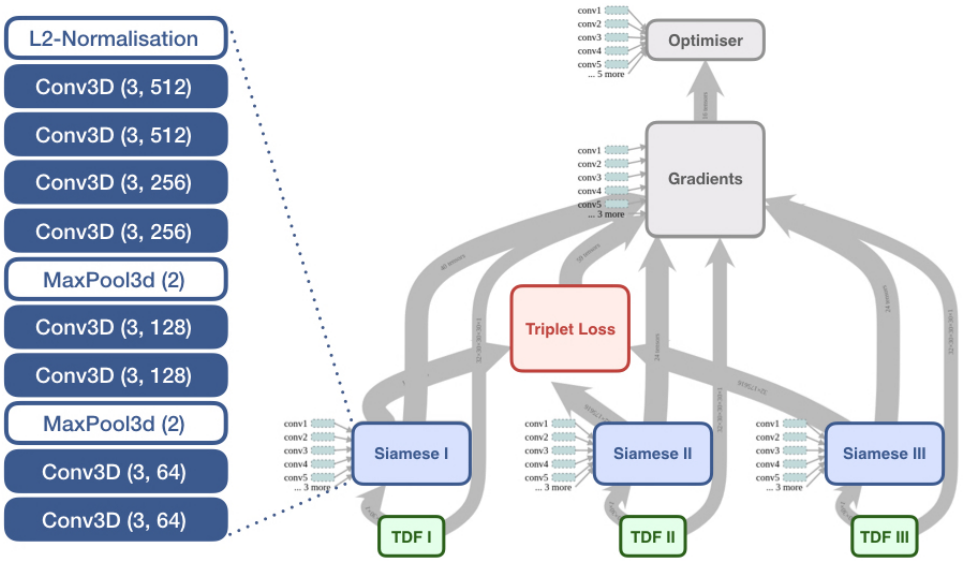

Figure 3.5: ***Network Architecture***

structure of the siamese branches (left-hand side). The branches consist of eight convolutional layers, each with kernels of shape 3^3, strides of 1 and the number of filters in the convolutions is increased from 64 in the first two layers to 512 in the last two, thus doubling the number of filters for every other convolution layer. All convolutional kernels are initialised with xavier uniform distributions and activated with a 'rectified linear unit'[25]. Max-Pooling layers are implemented after the second and fourth convolution to increase robustness and decrease memory requirements. Finally an L_2-normalisation layer is added after the output of the final convolutional layer, the resulting tensors are flattened and passed to the triplet loss node. In the loss node, the two pairwise euclidean distances between x_0 & x_+ and x_0 & x_- are computed. The resulting arrays represent the distances in the embedding space for every match- and non-match-pair in the batch (multiplied with $1e+04$ to enforce $dist^2 > dist$). At this point online sampling approach may be utilised [26]. However these strategies are only to some extend applicable to the problem at hand due to considerable memory requirements of the volumetric TDF descriptors and the deep convolutional network. Hence, batch sizes with the used system (*11GB TitanX GPU*) were limited to 64 which limits the capabilities of online sampling approaches. Furthermore, the loss was implemented in such a way that for non-matches only the mean of the samples below the margin was considered to ensure that every sample below the threshold is penalised. The optimiser used was a *Tensorflow* implementation of an *Adam-Optimiser* which proved to produce better results than stochastic gradient descent.

3.3.1 Loss Variations

As previously mentioned, the pairwise euclidean distances offer the opportunity to process sampled batches before forwarding them along the network graph. In addition to the discussed loss function where the means of matches and non-matches are optimised, a strategy was implemented that identifies the 'hardest', i.e. most extreme samples within a given batch and only consider these for the loss and back-propagation. For this purpose, the distances are sorted and only a certain fraction α of e.g. $\frac{1}{2}$ *to* $\frac{1}{8}$ of the samples are chosen while the rest will be ignored. Apart from this strategic sampling, the rest of the loss function remains as known. Further-

more, due to observations presented in subsequent sections, a contrastive loss with an additional penalty on variance is tested. Given match and non-match distance distributions within a batch G_+ and G_-, this loss is implemented as:

$$L_{Var} = mean(G_+)^2 + \{\max(m_{tresh} - mean(G_-), 0)\}^2 + var(G_+)^2 + var(G_-)^2 \quad (3.1)$$

$$L_{Var} = L_{Contrastive} + var(G_+)^2 + var(G_-)^2 \quad (3.2)$$

3.4 Metrics

For the purpose of quantifying the performance of the network, an error rate was chosen such that only false matches are considered. This is due to the requirements of the reconstruction process where non-match predictions are not valuable as opposed to the matches. As for many other comparable approaches where predictions are made based on some sort of threshold, the choice of this threshold has a crucial effect on the error rate. This becomes intuitively clear looking at figure 4.7 which shows the distribution of match- and non-match distances in the embedding at two different points during training. Figure 4.13 shows this relation for the earlier of the two. Hence the error rate at 95% recall is considered for validation in addition to analysis on the effect of different decision rules. Recall, error rate and needed respective definitions are outlined below:

		Prediction: Match	Prediction: NonMatch
Truth	Match	**TP**	**FN**
Truth	NonMatch	**FP**	**TN**

$$Recall = \frac{\#\ True\ Positives}{\#\ True\ Positives + \#\ False\ Positives} \quad (3.3)$$

$$Error\ Rate = \frac{\#\ False\ Positives}{\#\ False\ Positives + \#\ True\ Negative} \widehat{=} \frac{\#\ False\ Positives}{Constant} \quad (3.4)$$

3.5 Reconstruction Pipeline

In a final step, the network's capabilities concerning the actual reconstruction are evaluated. The crucial aspect of the reconstruction process is the identification of a sufficient number of correspondences with minimal errors and outliers:

3.5.1 Correspondence Identification

Two subsequent images of a sequence are chosen. Through segmentation and distance constraints, the parts of the image which do not display any tree parts are cut out and up to 20,000 of the remaining 'tree-pixels' are randomly selected (Some images contain less than 20,000 tree pixels, in these cases all of the existing points

are sampled). Hence, for each frame approximately 20,000 TDFs are generated and random combinations of TDF pairs from both frames are fed through the siamese networks to compute the euclidean distances in the embedding. For the work presented, 128,000 of these TDF combinations were sampled for a given frame combination. Combinations with descriptor distances below some specified threshold or decision rule are considered correspondences and will be used in subsequent processing steps.

3.5.2 Reconstruction Approach

The output of the correspondence identification are two point clouds (P_A & P_B) which each represent the correspondences identified with respect to their respective camera position. Since, theoretically these points represent the exact same points on the tree of interest, alignment of both clouds in terms of rotation and translation gives an estimate of the relative rotation and translation of the camera in-between individual frames. There exists a wide range of algorithms of varying complexity to this accomplish this alignment. Due to time constraints and issues with the distribution of correspondences, a rather simple RANSAC + SVD approach was used. The respective centroids of the point clouds are centered and subsequently a singular value decomposition is computed for the covariance matrix (H). From there, the rotation and translation can be computed in a straight-forward manner.

Computation of the covariance of the centered point clouds:

$$H = \sum_{i=1}^{N} (P_A^i - \frac{1}{N} \sum_{i=1}^{N} P_A^i) \cdot (P_B^i - \frac{1}{N} \sum_{i=1}^{N} P_B^i)^T \tag{3.5}$$

Singular value decomposition is performed on the covariance matrix H:

$$H = U \Sigma V^* \tag{3.6}$$

and the rotation matrix is then given as:

$$R = V U^T \tag{3.7}$$

With the acquired rotation, the corresponding translation is given as:

$$T = -R \cdot \frac{1}{N} \sum_{i=1}^{N} P_A^i + \frac{1}{N} \sum_{i=1}^{N} P_B^i \tag{3.8}$$

This approach to rotation and translation estimation begins to fail rather quickly with increasing numbers of errors and outliers in the correspondence clouds. Thus RANSAC was applied to cope with the outliers and find a good fit.

Chapter 4

Results

4.1 Data Generation

In order to be able to easily generate training data from existing RGBD sequences and control respective parameters, a data generation pipeline had to be implemented.

4.1.1 State and Limitations

At the current state, given any RGBD image sequences and corresponding poses, the system is capable of generating arbitrary numbers of training samples. The parameters that can be adjusted are the size and resolution of the extracted volumetric patches, minimum- and maximum- distances between the three points x_0, x_+, x_- as well as relative angle and position constraints for the RGBD drone.

A limiting factor is a considerable computation time and, particularly for GPU-utilisation, considerable memory requirements. Therefore, the work and results presented have been computed using pre-computed training data instead of online mining of samples. Though online sampling straight from RBGD sequences is possible (especially utilising cluster or cloud- computation resources), there was no indication that the number of training samples used was insufficient for the task.

4.2 Training

The network was trained on 14,000 TDF triplets with a validation set of 2,000 triplets from the same distribution. The batch size was set to 32 to reduce computation times and memory requirements. Training progress is depicted in figure 4.2 and 4.1. As previously mentioned, the loss function is composed as sum of two individual losses, penalising matches and non-matches, respectively. The loss exhibits considerable noise which to some extent, particularly for the match-loss may be explained with a relatively small batch size. Concerning the non-match part of the loss function, the governing reason for the high amount of noise is the fact that the loss drops to zero in case of no below-threshold-samples within a given batch. Furthermore, since non-matches above threshold are ignored, this part of the loss is mostly composed of few, extreme samples which contributes to a non-smooth loss progression. However, the overall loss is steadily decreasing and likewise, the means of matches and non-matches are gradually moving apart from each other thus behaving as expected (apart from the discussed noise) from the loss function and its intuition as explained in previous sections. Training progression and metrics

for different sampling strategies and loss variations are presented in the following.

4.2.1 Standard Contrastive Loss

Firstly, the training progression for the 'standard' contrastive loss as previously discussed is presented. The means are computed over the entire batch of each iteration for match- and non-match-distances, respectively. Progression of means of the two distributions are depicted in figure 4.1 and the corresponding loss is shown in figure 4.2. The weights of the convolutional kernels of each layer are shown in the histogram plots of figures 4.3 and 4.4 for different stages during the training. The tested loss and sampling-strategy-variations had no significant effect on the distribution of the convolutional kernels.

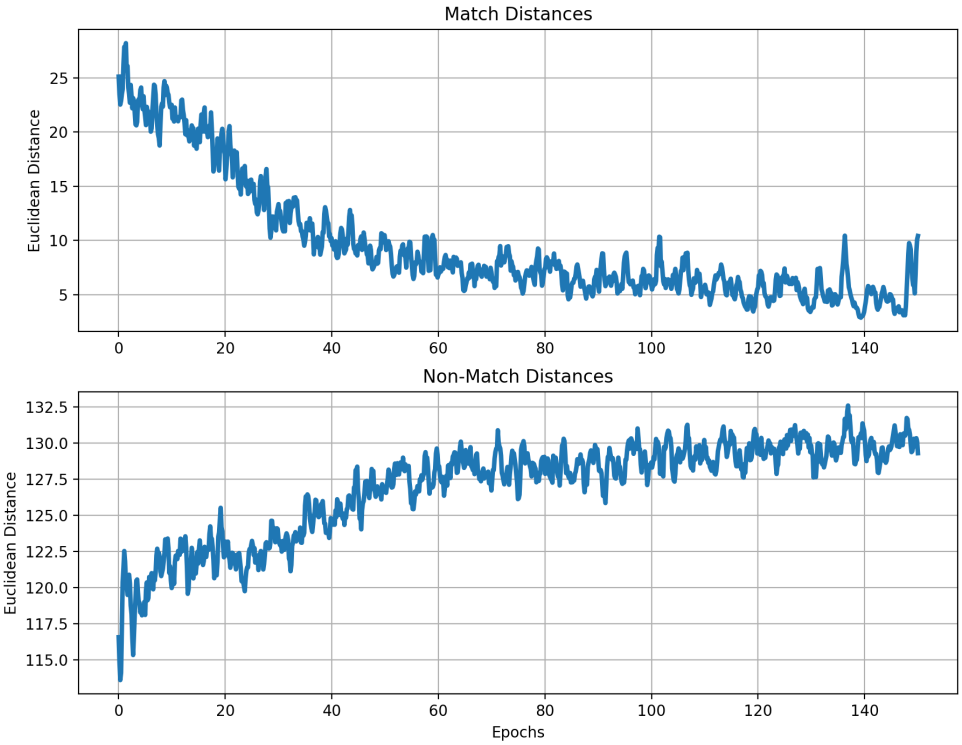

Figure 4.1: ***Training progression of respective distances for normal implementation of contrastive loss without any processing.*** *As expected from the discussed intuition of the contrastive loss, the respective means of both distributions move apart from each other. The distance between the both averages is increase by* 40% *over the presented 150 epochs of training.*

The resulting weight distributions in the convolutional kernels are depicted in figure 4.3. Regardless of any hyper parameter tuning that has been tested over the course of this project, the weights always exhibited the same general behaviour. Within a relatively small number of iterations (∼1-3 epoch), the distribution of weights in the respective convolutional layers evolves from its 'Xavier'-uniform initialisation to distinctive spikes close to zero, thus implying favourably sparse kernels as well as a learning process that has (almost) converged since the layers are fully 'specialised' at this point. Also, the number of iterations needed for the convolutional weights to arrive at this 'converged' state is correlated with the position in the graph of

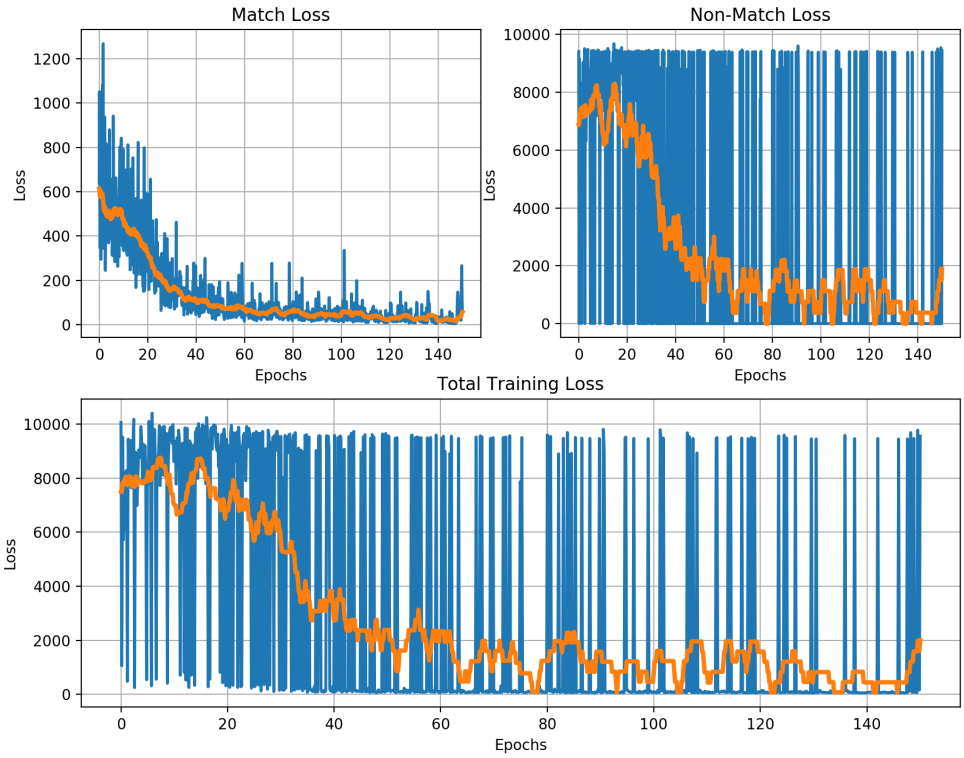

Figure 4.2: ***Total loss and individual losses (blue) with running average (orange).*** *Both losses decrease significantly within the first* ~ 40 *epochs. However, especially the non-match loss (up,right) exhibits considerable noise which is mostly contributed to the fact that the non-match loss drops to zero if all considered samples are above the threshold margin.*

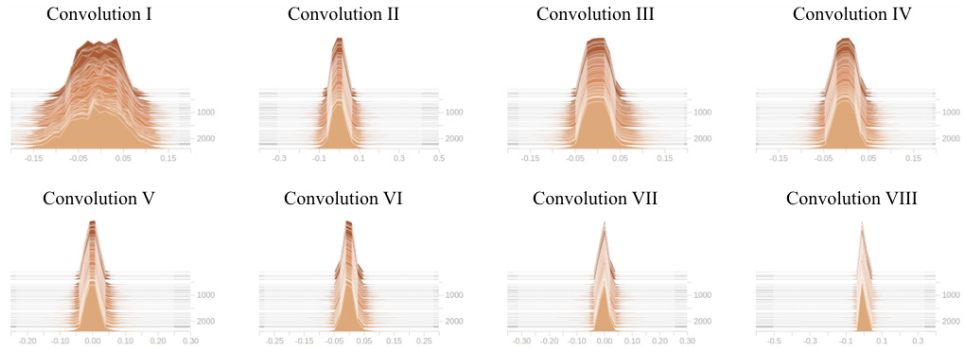

Figure 4.3: ***Distribution of Weights in the Convolutional Kernels after*** $\sim 2,000$ ***iterations.*** *In general, the further down the forward propagation graph, the quicker the convergence. Initialised with a xavier-uniform distribution.*

the respective convolution. The kernels in the more shallow layers of the network arrive at a slightly wider distribution of weights and tend to take considerably more iterations, i.e. training time to converge.

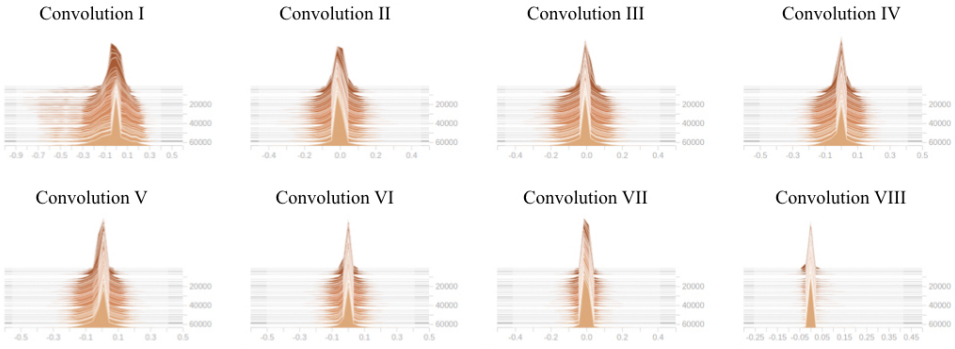

Figure 4.4: ***Distribution of Figure 4.3 after*** $\sim 60,000$ ***iterations.*** *The weight distributions of all convolutional layers converge to distinctive spikes close to zero. The early layers tend to arrive at wider distribution and require considerably more iterations as the layers further down the graph.*

4.2.2 Online Sampling & Variance Penalty

As previously discussed, the pairwise euclidean distances acquired from the node that connects the outputs of the individual siamese nets, provide the option of pre-processing or sampling before forwarding them to the loss and back-propagation. The evaluated approach sorts the euclidean distances and chooses a specified fraction α, consisting of four equally sized parts: Matches and non-matches for leaves and non-leaves, respectively. For each of these, the most extreme, i.e. 'hardest' samples are chosen. This way the mean of the worst fraction α of given samples is optimised as opposed to the entire batch mean. Presented values for α : 0.5, 0.125. With the used batch size of 32 and e.g. $\alpha = 0.125$, this results in 8 evaluated pairwise distances (32 for $\alpha = 0.5$).

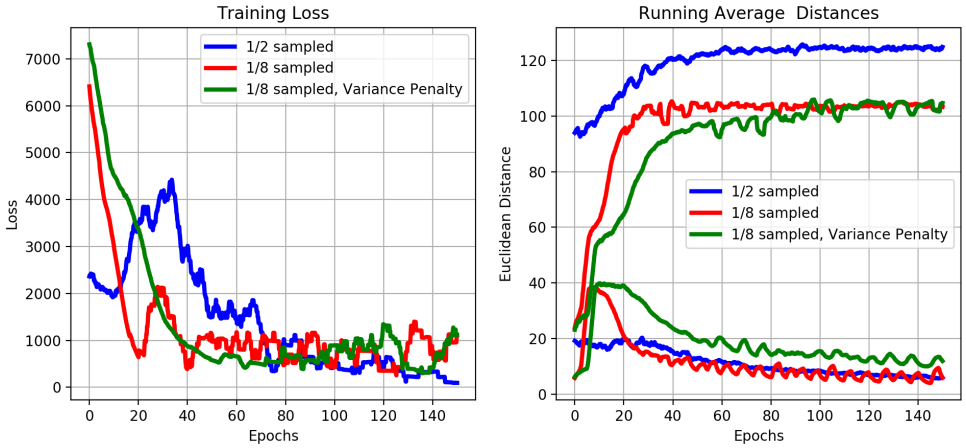

Figure 4.5: ***Training loss (left) and resulting distance distribution means (running average, right).*** *The plotted distances are the means of the distances considered for loss computation for matches and non-matches, respectively. (Hence two plots per colour on the right figure).*

Both losses and distance means behave similar for all tested setting as for the standard approach. For the standard (benchmark) implementation, the distance means move more gradually and experience less relative change, however this is mostly

due to the fact that the entire batch is considered. The most notable difference observed during training is the reduction of noise in the loss. This effect is noticeable when comparing the loss for a test with 1/8 sampling and variance penalty in figure 4.6 to the loss of the standard loss in figure 4.2. The respective performances on validation data are presented in the following section.

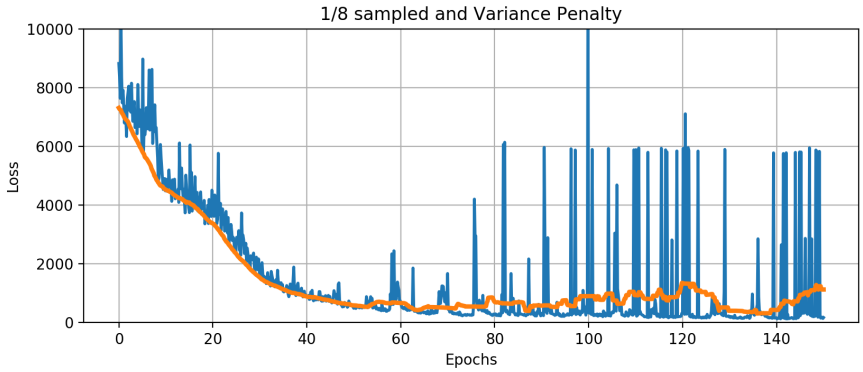

Figure 4.6: ***Training loss for a bias toward hard samples and penalised variance*** *(sampled fraction: 1/8). The loss exhibits considerably less noise than for the standard implementation depicted in figure 4.2.*

4.3 Validation

The performance of the network was monitored through frequent evaluation of a validation set consisting of 4.000 point pairs (2.000 matches & 2.000 non-matches). Results acquired on the validation set are presented in the following. For this purpose, the validation set was sampled every 200 iterations.

For the standard implementation of the network, the progression is shown in figure 4.7. The upper plot shows the normalised values of the loss and its individual parts as well as the error rate measured at 95% recall. The first observation is a very quick decline of the error rate to below 0.2% in almost all tests within the first few epochs. The corresponding distribution of the validation set is depicted on the bottom left figure. The 4 colours represent leaf matches (green), leaf non-matches (red), non-leaf matches (blue) and non-leaf non-matches (magenta). There is no considerable difference in the performance between leaves and non-leaves which might be due to the fact that trunk-, branch- and twig-performances 'cancel' each other out such that the total distribution is very close to the leaves performance. This was indicated by tests on older data sets which, in contrast to current data, had this distinction in their segmentation. After $\sim$ 20 epochs, the validation error starts to diverge to $\sim$ 3% over the depicted range and keeps on diverging afterwards. However, there is no indication or corresponding reaction in the validation loss. The same distribution of the validation samples is depicted on the bottom right for a later point during training. Particularly for the non-match distribution, a move of the mean toward the upper boundary is noticeable as predicted from the loss objective, although accompanied by an increasing validation error. This is mostly due to the effect of the increasing variance shown in figure 4.8 which results in an increased amount of outliers. The relationship between recall and error rate is depicted in figure 4.13. Even a slight increase of outliers can have considerable impact on the achieved error rate.

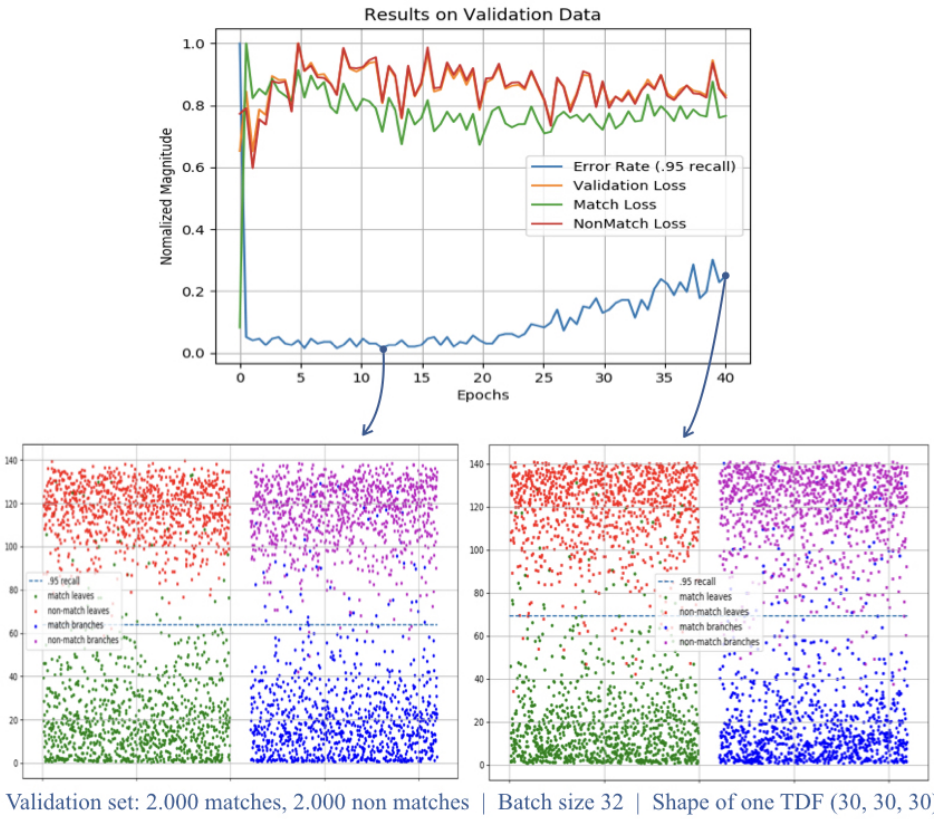

Figure 4.7: ***Normalised Progression of Error Rate and Validation Loss with corresponding distance distributions.***. *At first, within a few hundred iterations, the error drops from* $\sim 10\%$ *to below* $\sim 0.2\%$. *After* $\sim$ *20 epochs, the error rate on the validation set starts to diverge. However the validation loss does not reflect this. The reason lies in the different difference in the objective of the loss function and the measured error rate. The left scatter plot shows the distribution after twelve epochs with an error at* 95% *recall of* 0.1% *while the right plot after 40 epochs has an error of* $\sim 3\%$

A comparison of the evaluated approaches is given in figure 4.9 which shows the error rate at 95% recall. As visible, the general characteristics of the error rate progression remain the same for all tested variants. The error very quickly converges to excellent values below 0.2% with the first few iterations. However, after $\sim$ 20 epochs, the validation error starts diverging. The extent to which it diverges could be drastically reduced using alternative implementations of the loss and sampling strategies. This poses the question of whether there is any incentive to keep training beyond 10 to 20 epochs which will be discussed in some more detail in later sections. The effect of the choice of threshold was also investigated to ensure that the error divergence is not due to the 95% recall margin. Two decision rules were tested for that purpose: $Threshold = 0.5 \times max(distances)$ and $Threshold = 0.5 \times training\ margin$. Neither of these changed the characteristics notably, although the latter enforces a decision rule which after a few epochs achieves even better error rates since it set the threshold below the 95% recall margin.

The validation losses for one evaluated approach in comparison to the standard

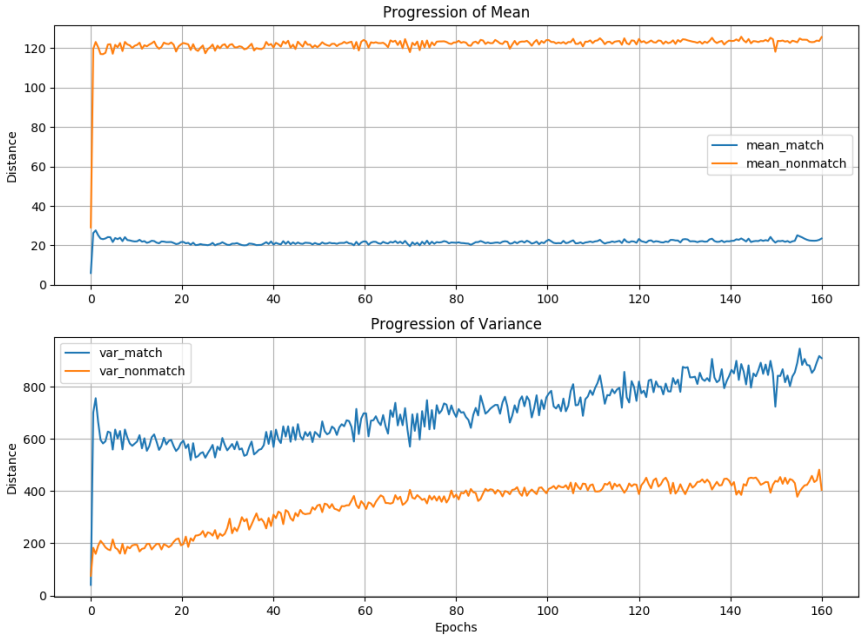

Figure 4.8: ***Mean- (top) and variance- (bottom) progression.*** *The slow and gradual increase of the distance between the two means happens at the cost of significantly increasing variances. As a consequence, the validation error starts diverging without any increase in validation loss.*

Figure 4.9: ***Error rate measured at*** 95% ***recall.*** *All of the evaluated variances exhibit the same general behaviour of very early convergence to excellent error rates of mostly* $> 0.1\%$ *with the first few iterations, mostly reaching their optimum at* ~ 20 *epochs and subsequent (also rather early) divergence. However the extent to which the error diverges could be drastically reduced using alternative sampling strategies and loss variations.*

implementation is shown in figure 4.10 while figure 4.11 shows the validation losses for all of the discussed approaches. In contrast to the standard implementation, the pre-sampled approaches exhibit a rise in validation loss corresponding to the increase of the validation error. Harsher sampling (i.e. smaller fraction α of samples considered) also results in a more distinct increase of the loss. The simultaneous increase of validation -loss and -error implies a favourably increased correlation between the objective optimised from the loss function and the performance measured by the error rate. However the error rate still diverges at some point with no clear incentive to keep on training beyond that. The distribution of the validation samples for penalised variances is depicted in figure 4.12.

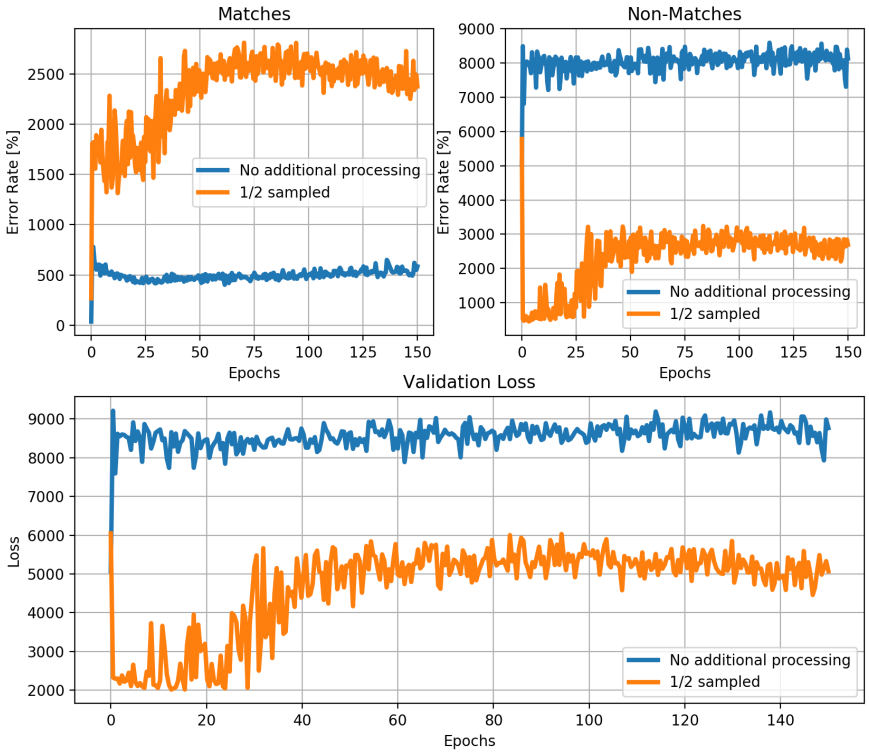

Figure 4.10: ***Total validation loss and individual losses for standard contrastive loss (blue) and an implementation with bias towards hard examples.*** *The loss function is stronger correlated to the error rate as for the standard implementation as visible by the divergence of the orange plot, corresponding to the validation error divergence*

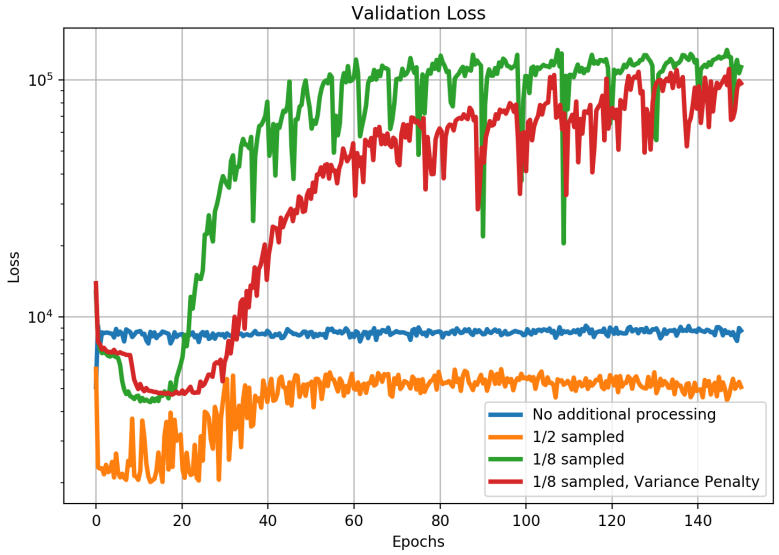

Figure 4.11: ***Comparison of validation losses.*** *The extent to which the losses reflect the validation error divergence depends on the applied degree of sampling. The stronger divergence of the validation losses suggest a stronger correlation of the loss to the error rate when using sampling strategies as discussed.*

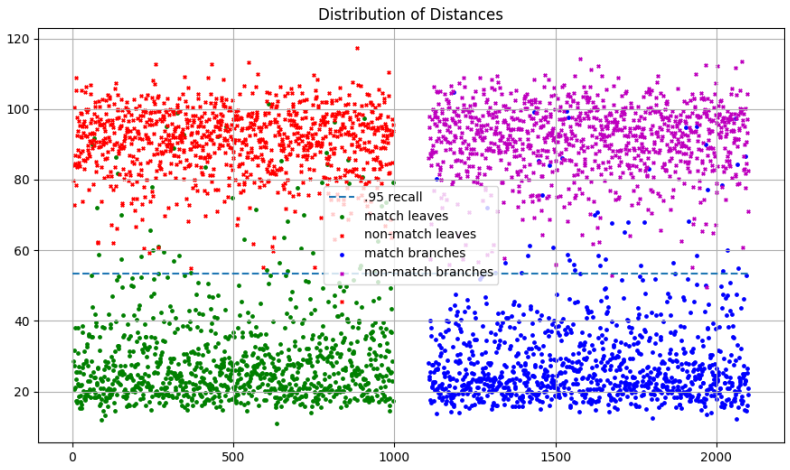

Figure 4.12: ***Distribution of distances in the embedding for an implemented penalty of variance.*** *A denser distribution of the distances is achieved at the cost of decreasing distance between the respective means.*

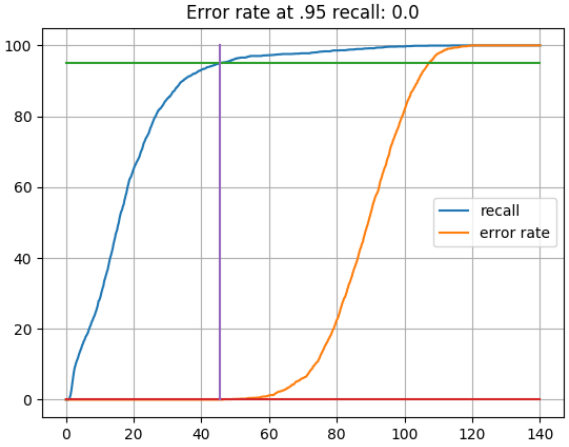

Figure 4.13: ***Recall - Error Rate Relationship.*** *For this particular point during the training, the error measured on the validation set at* 95% *recall was* 0.0%*. A (slight) increase of outliers pushes the intersection of the recall plot with the* 95% *line to the right while simultaneously flattening the error plot, thus moving its point of divergence from* ~ 60 *at this point to the left. Consequently, the choice of decision rule as well as the amount of outliers have a crucial effect on the measured error rate.*

4.4 Reconstruction

The trained network is used for the identification of correspondences in any given combination of RGBD frames. The approach was tested on the two consecutive frames depicted in figures 4.15 and 4.16 and respective results are presented in the following section.

4.4.1 Correspondence Identification

As previously mentioned, the process of sampling for correspondences begins by feeding an arbitrary number of random TDF patch combinations through the siamese networks and compare the euclidean distances in the resulting embedding. The result can be seen in figure 4.14. The sorted embedding distances for 128,000 random combinations are plotted on the left side of the graphic. The first few hundred distances close to zero stem from defects at the upper boarders of the frames. These will be ignored. For the remaining point combinations, a threshold is set to act as decision rule. In figure 4.14, this threshold is indicated by the orange, dotted line and was set to a value of 40. Thus considerably below e.g. the 95% recall threshold. The resulting collection of pairwise distance is plotted on the right side of figure 4.14. For the used setting of 128,000 sampled point combinations and a relatively low threshold, this results in 161 correspondences for this example.

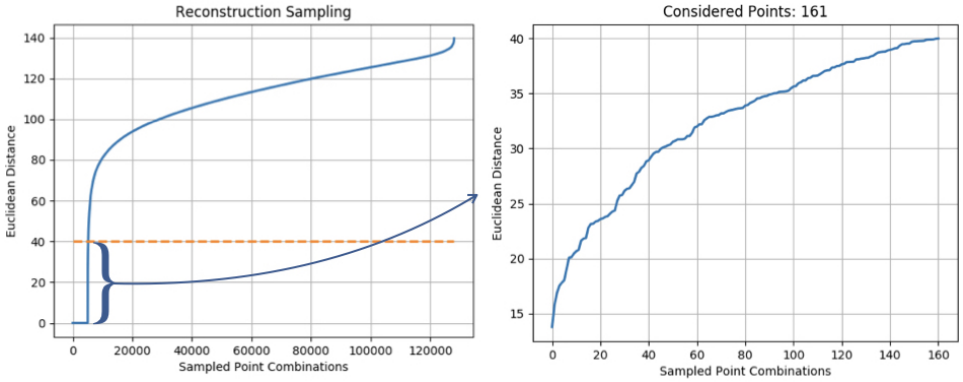

Figure 4.14: ***Pairwise euclidean distances for the sampled point combinations (L) and distances of considered points below threshold (R)***

The frames that were used in this experiment as well as the correspondences identified by the network are shown in figures 4.15 and 4.16. The latter shows all of the 161 predicted correspondences while the former only shows some sub-sample of the point combinations with the lowest pairwise euclidean distances in the network's embedding. While there are some obvious errors visible particularly in figure 4.16, the majority of predicted matches seems to be reasonable. However, as can be seen on aforementioned figures, the correspondences are far from evenly distributed over the tree. This is due to the fact that the leaves of the tree exhibit much more occlusions than the branches and trunks and poses a challenge for the reconstruction since particularly the estimation of the relative rotation suffers if the majority of points are along one axis.

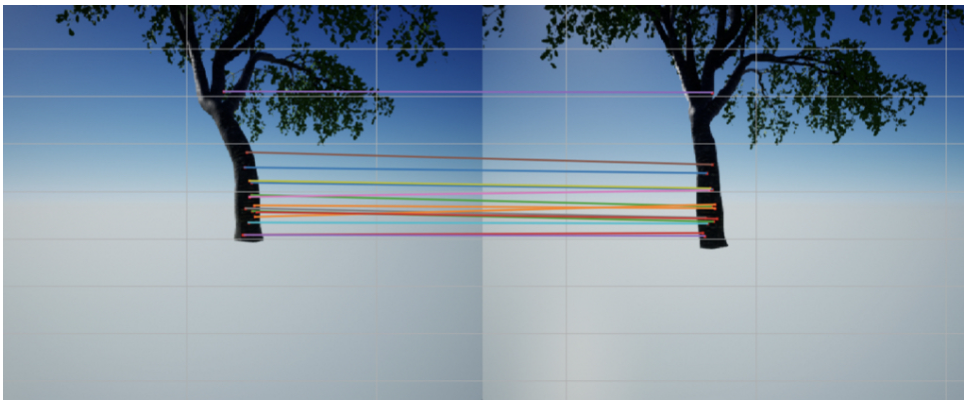

Figure 4.15: ***Sub-sample of correspondences with lowest pairwise distances in the descriptor embedding***

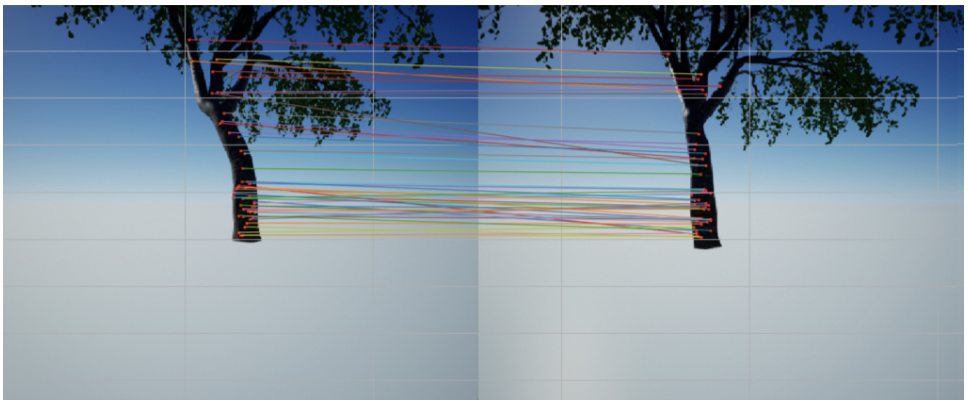

Figure 4.16: ***Resulting correspondences from sampling of figure 4.14.*** *Occasional harsh defects are identifiable by the slope of the connection lines. The* 161 *points with the lowest similarity measures were chosen for this figure. Unfortunately, the identified correspondences are poorly, i.e. unevenly distributed over the tree. This effect is only increased if the decision threshold is lowered as visible in figure 4.15. The fact that almost all correspondences are positioned along the axis of the trunk makes the reconstruction prone to errors, particularly considering the rotation part.*

4.4.2 Correspondence Alignment

Alignment of the resulting correspondence point clouds was performed using a singular value decomposition (SVD) approach combined with RANSAC to try to cope with outliers and wrong correspondences. The resulting transformation heavily depended on the applied decision rule, i.e. threshold applied on the sampling shown in figure 4.14. This parameter controls the number of points that are used for a given number of sampled TDF combinations as well as the average 'quality' of the identified points. This effect can also be seen in figures 4.15 (lower threshold) and 4.16 (higher threshold, same as orange line in figure 4.14). A resulting reconstruction is depicted in figure 4.17. The error of SVD on the full correspondence cloud can be reduced by $\sim 80\%$ using RANSAC, however there remains a considerable error. For the results of figure 4.14, the computed Euler angles of the transformation were $[-0.0710, -0.0025, 0.2377]$ compared to the ground truth of $[0.0, 0.0, 0.25]$. Thus, the result of the correspondence alignment at this point does not yet enable

reconstruction of an entire tree with satisfactory results. This is mostly due to two circumstances which could not be overcome in the time frame of this project. Firstly, more sophisticated and advanced algorithms should be utilised for the alignment process. The used SVD and RANSAC approach provides a good starting point but surely does not represent the most advanced method. Secondly and most importantly, the distribution of identified correspondences along the tree needs to be optimised. An initial step could be more extensive sampling to acquire more points off the central axis, however this will require considerable computational efforts and does not address the problem. Instead, the network needs to be encouraged to identify more leaves which might be accomplished through implementation of a bias toward leaf points during training or other approaches.

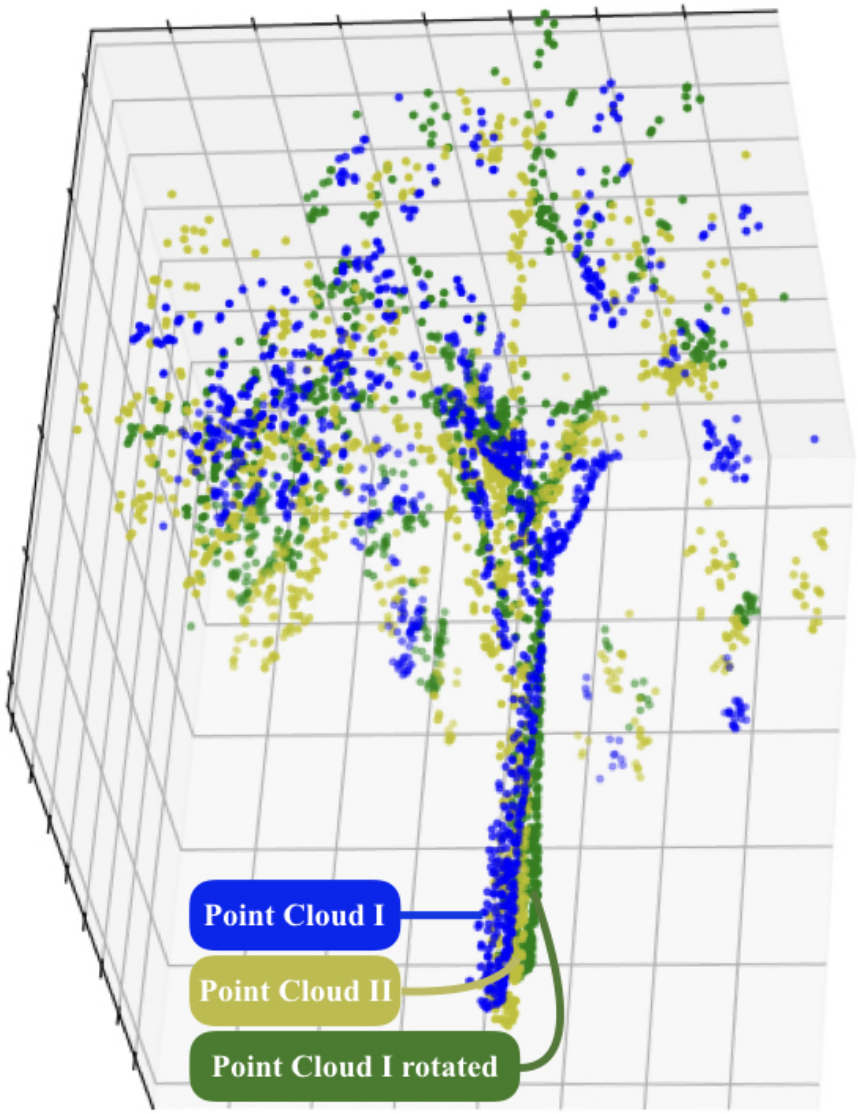

Figure 4.17: ***Alignment of correspondence point clouds.*** *The green and yellow point clouds represent the correspondences identified in the respective frames. Through the discussed SVD+RANSAC approach, the first (blue) point cloud should be rotated and translated such that it aligns with the green one. This transformed first (blue) point cloud is depicted in yellow. As visible, there is still a considerable error in the rotation estimation as the yellow point cloud is over-rotated.*

Chapter 5

Discussion & Conclusion

The '3DMatch' approach by Zeng et al. [1] was implemented and evaluated on tree point clouds for the task of correspondence identification. The current state and limitations will be briefly reviewed in addition to future work on the project:

Limitations

At the current point in time, the only feasible approach to developing and testing works like the one presented, is to utilise synthetically generated trees and respective RGBD images. Within this constraint, the motivation is to evaluate the discussed approach on synthetic data in order to lay a foundation for future work involving real data. Additionally, at the current state, no noise has been added to the acquired depth measurements or camera orientations and coordinates to study the systems robustness.

Resulting correspondences from sampling of subsequent frames exhibit very uneven distribution of correspondence points along the tree which poses a challenge for the reconstruction since the transformation becomes poorly defined if all anchor points are located along one central axis. More advanced alignment algorithms could be utilised to try to cope with this issue, however the governing factor to be optimised for reduced reconstruction errors is the distribution of correspondences along the tree.

State

At its current state, the system is capable of generating training instances from arbitrary RGBD sequences and training a convolutional siamese neural network to distinguish matches from non-matches and eventually identify correspondences. Excellent error rates of close to zero percent are achieved within ~ 10 epochs with generally quick convergences. The validation error starts diverging after ~ 20 epochs, regardless of evaluated network settings, however the degree of divergence could be considerably reduced through utilisation of alternative sampling strategies and penalisation of variances. Still there is no apparent incentive to train the network beyond approximately 20 epochs for either of the tested approaches. Divergence of the error without corresponding rise of the validation error could be explained with increasing variances and the correlation between loss function and error rate could be increased through implementation of sampling strategies. A pipeline for reconstruction has been implemented. However at the current state it suffers from a possibly sub-optimal choice of reconstruction algorithm and particularly a poor distribution of identified correspondences.

5.0.1 Future Work

Future work on the project should focus on incentivising or enforcing a wider distribution of identified correspondences over all geometrical areas (such as trunks, branches, leaves and twigs) as this would be both beneficial for the reconstruction as well as represent a more general descriptor since the descriptors learnt during this work apparently favour non-leaf parts of the tree, which does not reflect in the results acquired from training and validation. Additionally, real RGBD data sets should be utilised once available to validate the results acquired so far. Furthermore, the variations of the loss and sampling approach showed promising results and should be pursued further. Lastly, the algorithms used for reconstruction should be improved and the individual parts of the approach should be compared to benchmark comparisons with other state-of-the-art approaches to correspondence matching as well as three-dimensional reconstruction from depth images.

Bibliography

[1] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 199–208.

[2] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 6959–6968.

[3] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 433–449, 1999.

[4] I. Interactive Data Visualization. speedtree.

[5] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 433–449, 1999.

[6] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369.

[7] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *European conference on computer vision*. Springer, 2004, pp. 224–237.

[8] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3384–3391.

[9] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[10] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 808–816.

[11] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3d deep shape descriptor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2319–2328.

[12] K. Guo, D. Zou, and X. Chen, "3d mesh labeling via deep convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 1, p. 3, 2015.

[13] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[15] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[16] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[17] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] P. Baldi and Y. Chauvin, "Neural networks for fingerprint recognition," *Neural Computation*, vol. 5, no. 3, pp. 402–418, 1993.

[20] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[21] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[24] Y. Wang, "Analyzing imagery of vegetation using deep learning," 2017.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[26] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.