Match Correspondence in Vegetation Point Clouds using Deep Learning

Benedikt Dietz
Semester Project

Supervision: Tejaswi Digumarti
Dr. Cesar Cadena Lerma
Dr. Paul Beardsley
Prof. Dr. Roland Siegwart

Motivation & Context

Learning feature descriptors for vegetation correspondence matching

Vegetation is challenging for conventional approaches

- Many repeated elements
- Large amount of occlusions
- Uniformity of colour

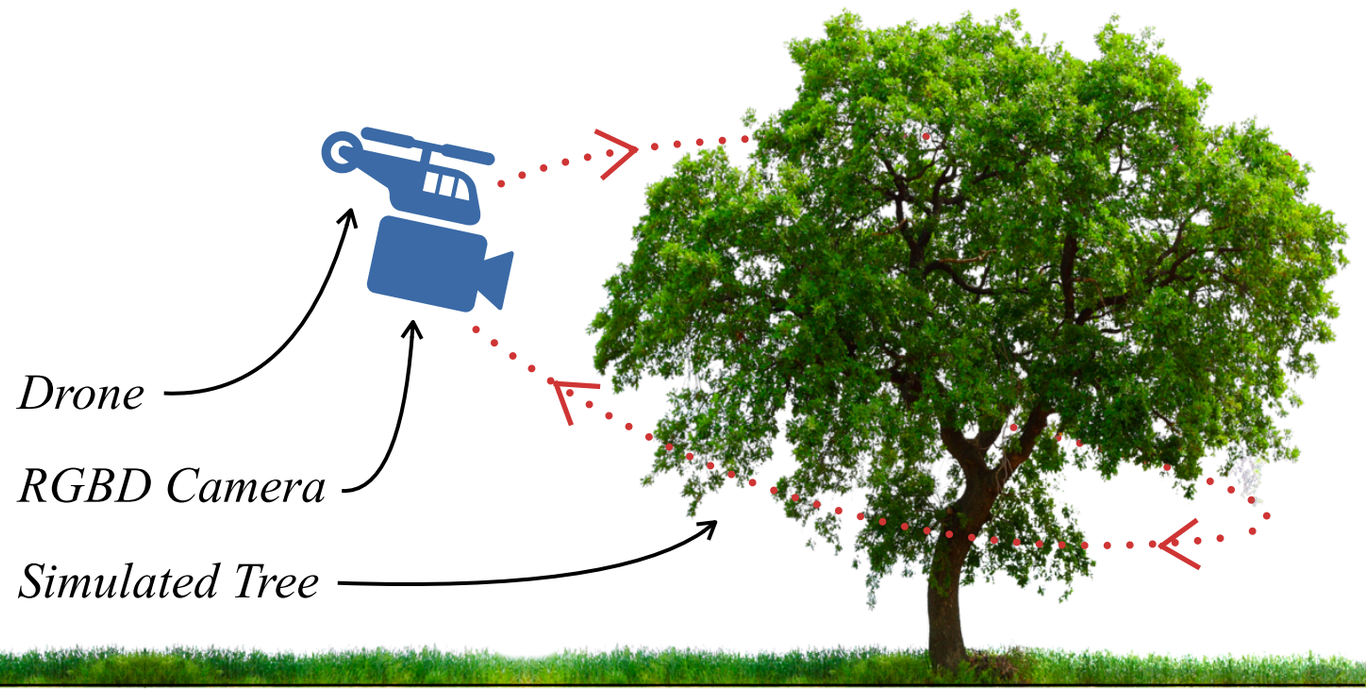
→

Result

- Nuisance in robotics applications
- Usually ignored using segmentation
- However interesting for entertainment industry

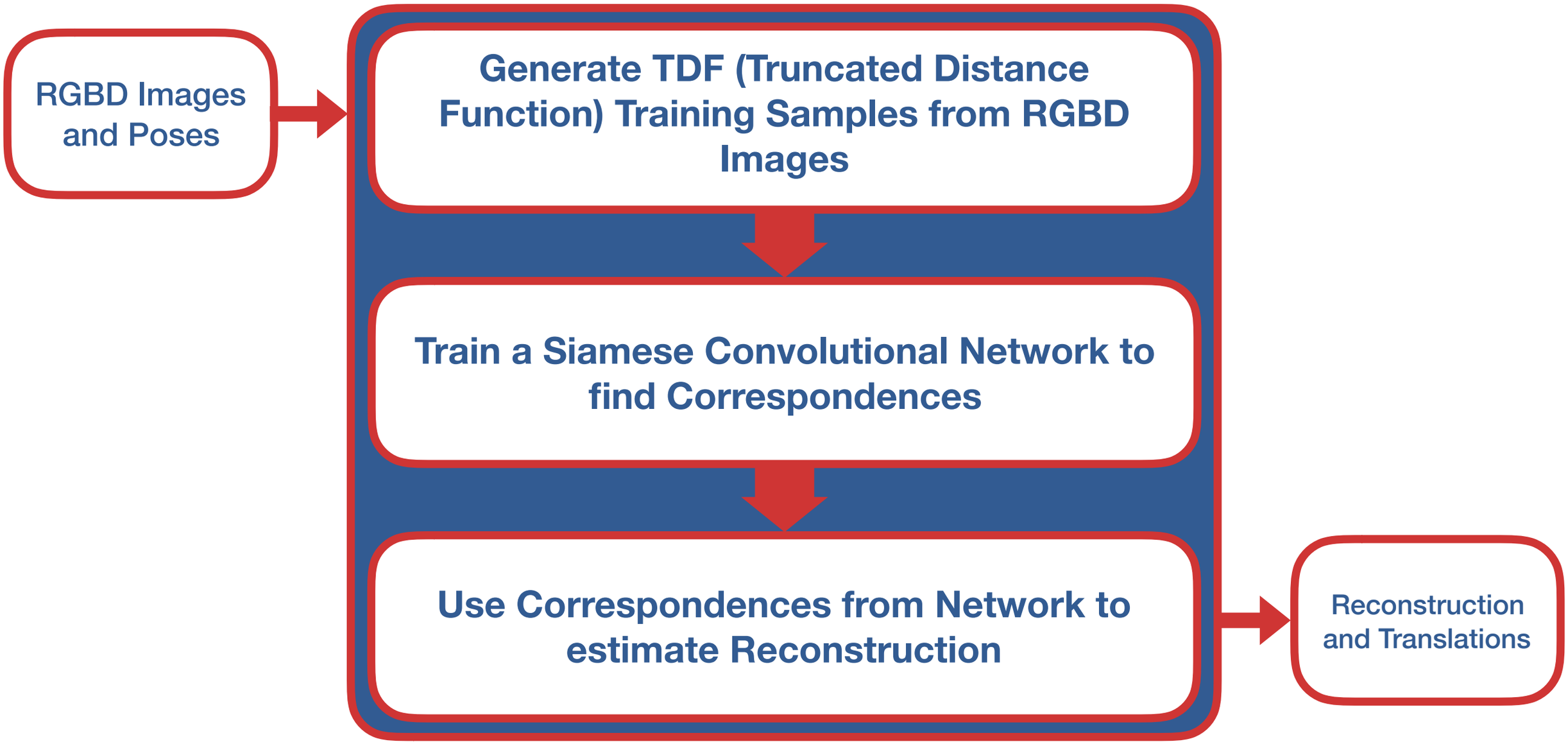
Motivation

- Recent success of 3DMatch publication using deep learning
  - Trained/ validated with indoor scenes
- Test/ validate/ optimise 3DMatch approach for veg. point clouds



Big Picture of the Project

The overall project and presentation is split into three parts:

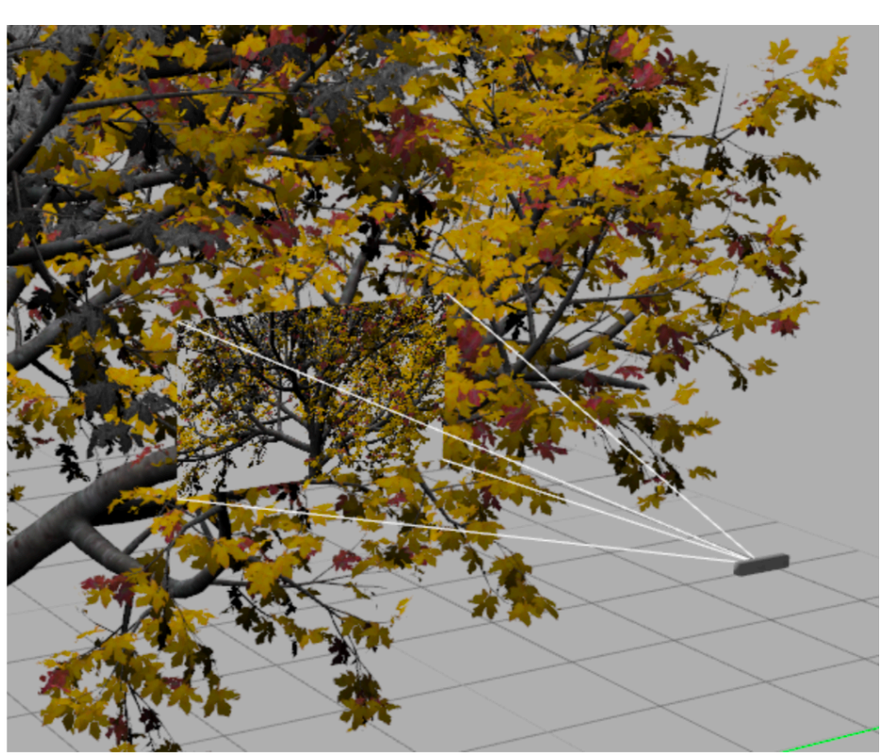


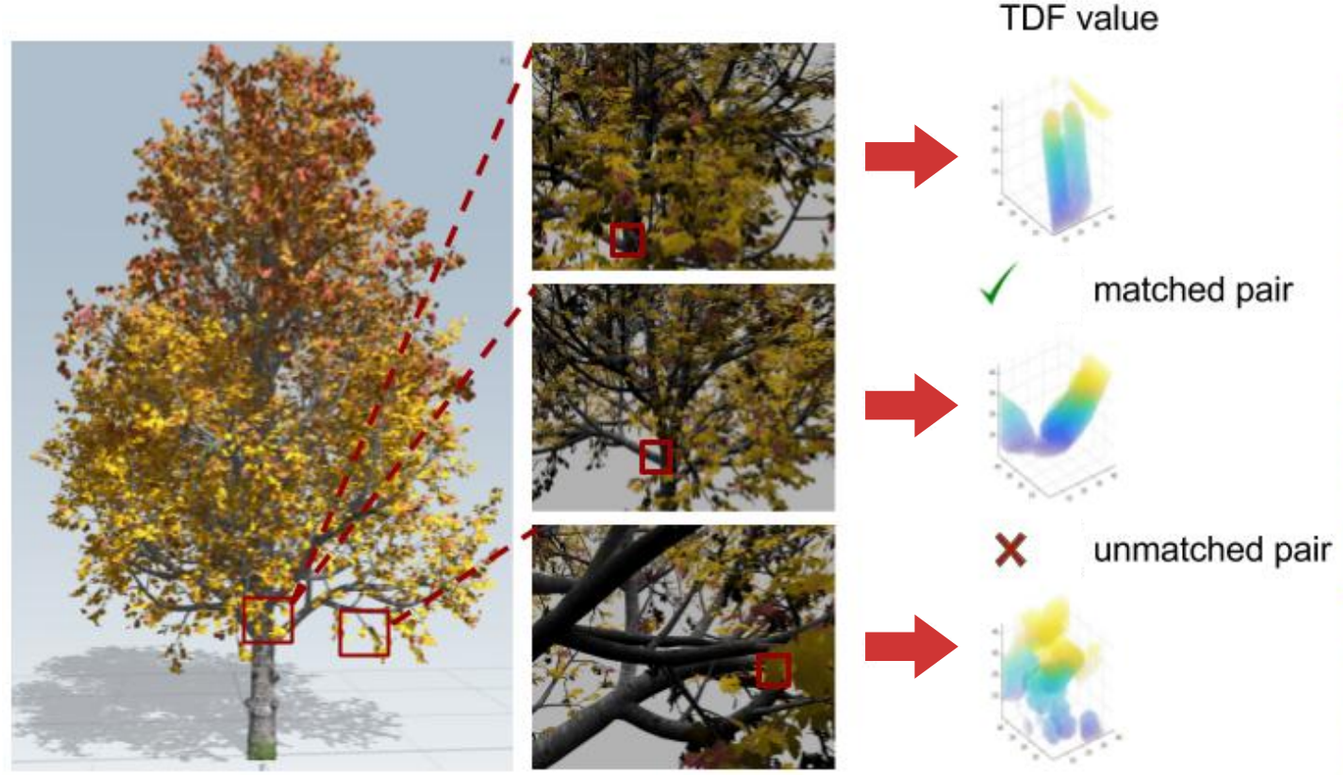
Data Production

From Images to TDF Training Data

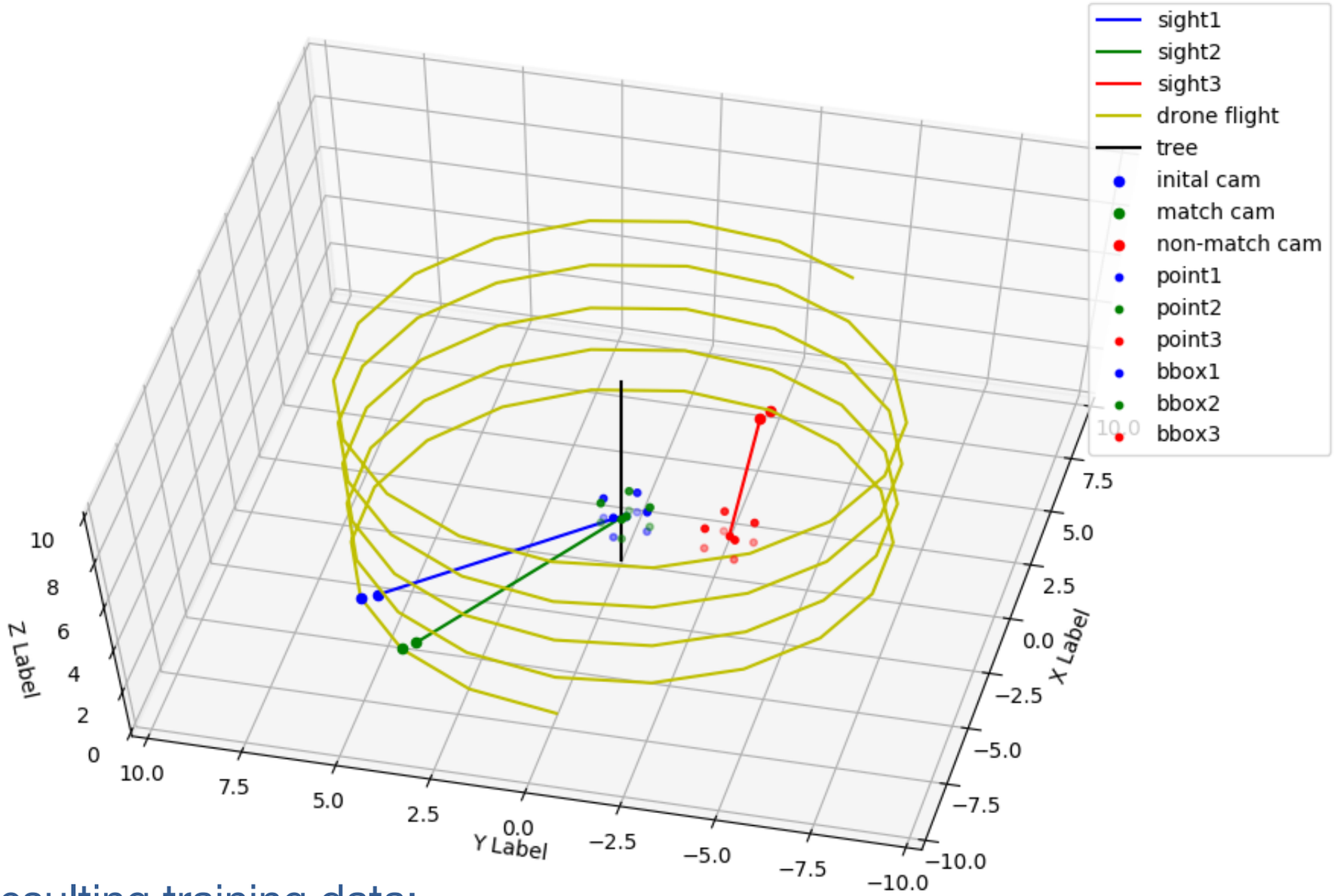
Due to a lack real of RGBD tree data sets, artificial trees and images were used

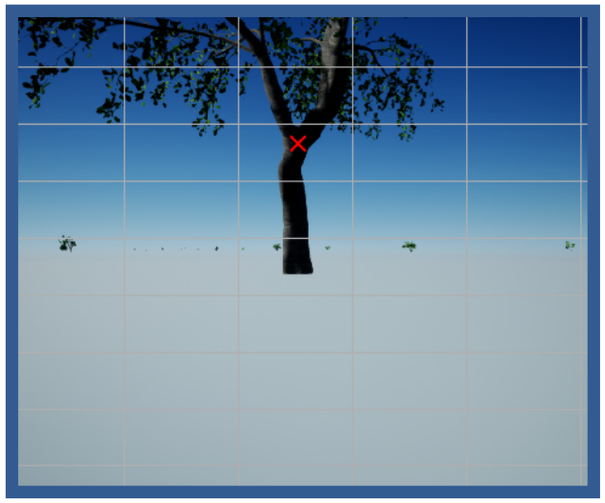
- ➡ Virtual drone flies around the tree on a specified trajectory
- ➡ Matching/ non-matching points on the tree are randomly chosen
- ➡ Certain volume (30x30x30 [cm]) around the specified point is extracted
- ➡ Point cloud is generated and converted into a TDF

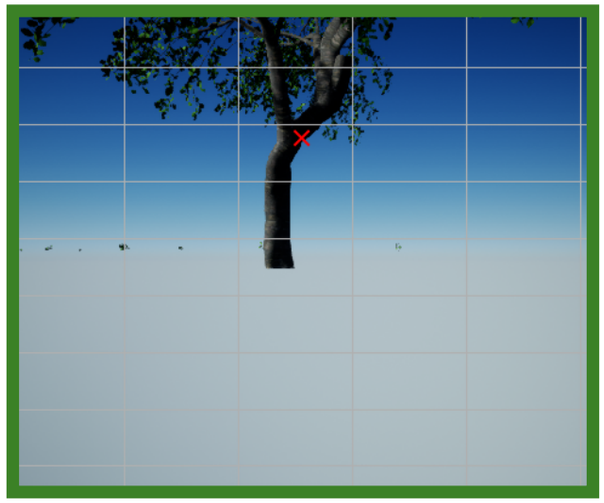




Data Generation







Resulting training data:

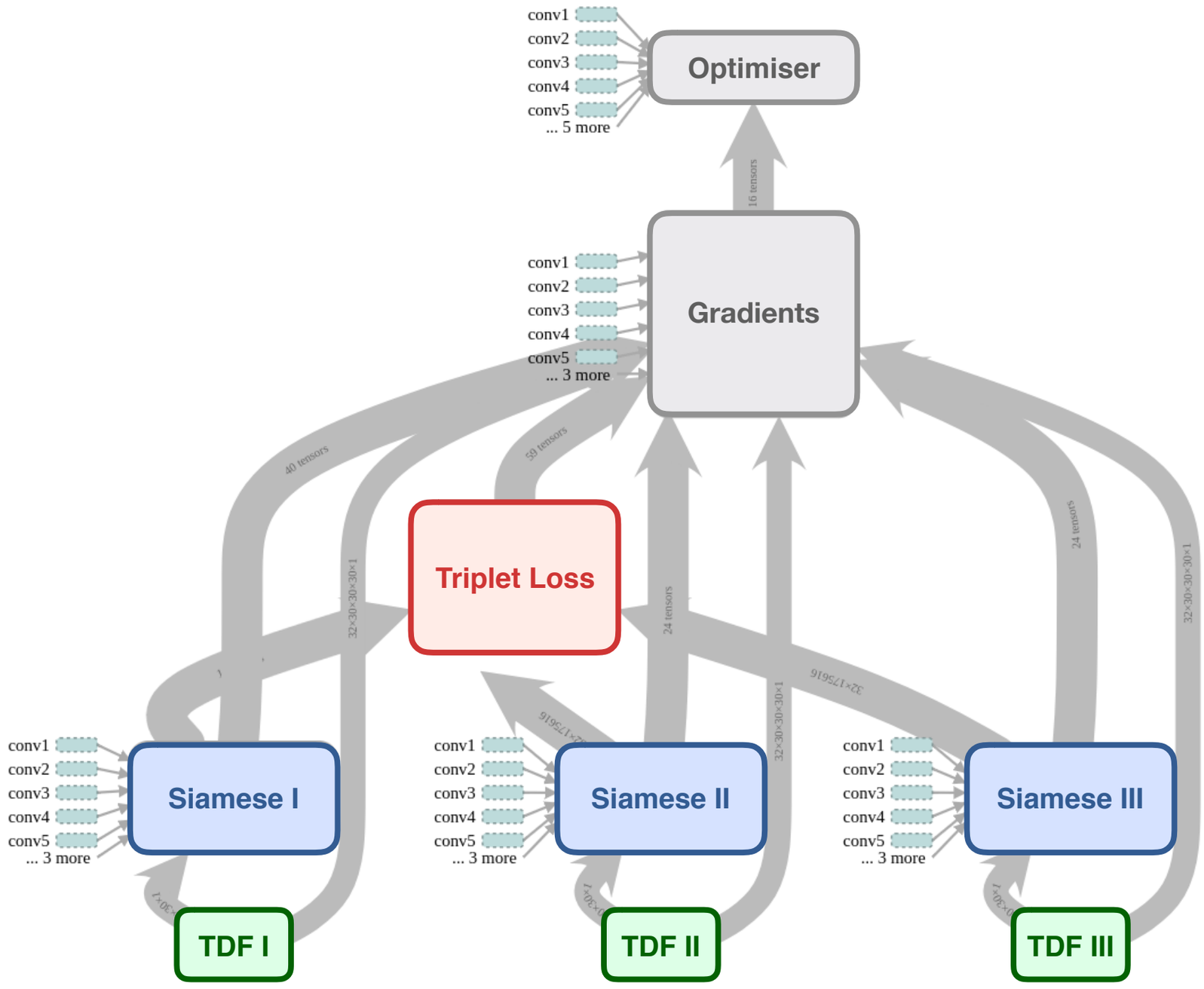
Triplets of TDFs (initial point | match | non-match)

3rd point is randomly chosen on the same tree with a min. distance

Siamese Network

Siamese Convolutional Network

- L2-Normalisation
- Conv3D (3, 512)
- Conv3D (3, 512)
- Conv3D (3, 256)
- Conv3D (3, 256)
- MaxPool3d (2)
- Conv3D (3, 128)
- Conv3D (3, 128)
- MaxPool3d (2)
- Conv3D (3, 64)
- Conv3D (3, 64)



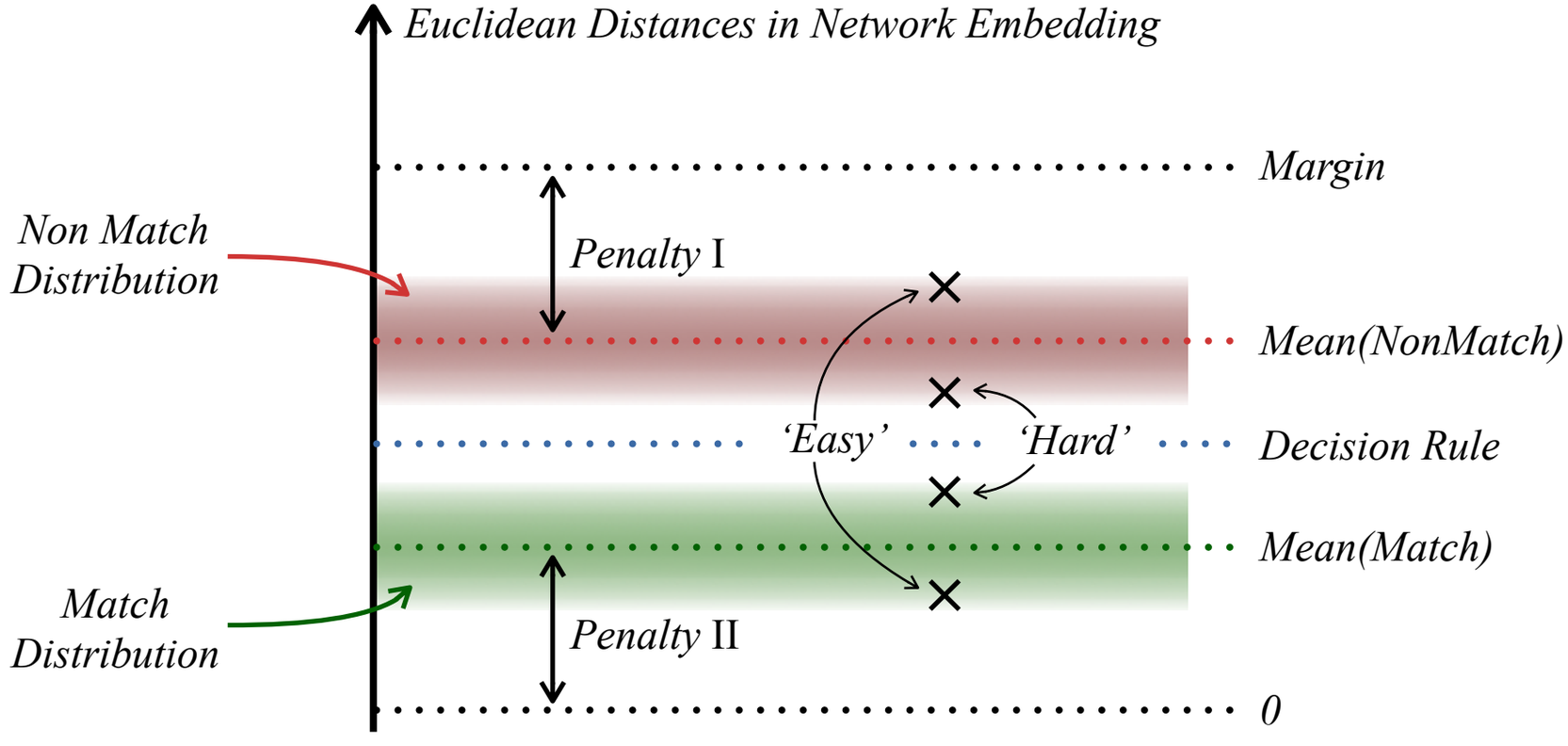
Contrastive / Triplet Loss

Intuition: Push means of match- and non-match-distances in the embedding apart

$$L_C = (Y-1)\, D_W^2 + Y\, \{\max(0,\, m - D_W)\}^2$$

L_C - Contrastive Loss | D_W - Euclidean Distance | Y - Label | m - Margin

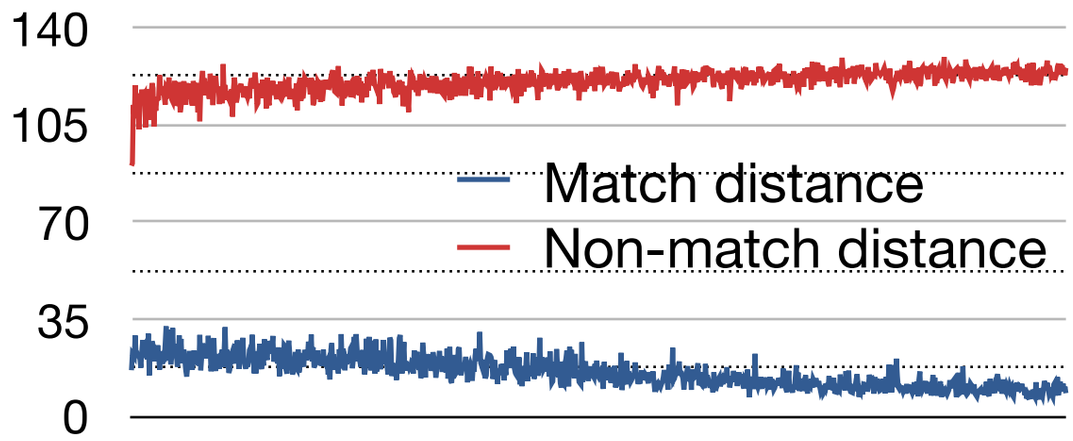
Alternative implementation: **Triplet Loss** (3 TDFs instead of 2 TDFs + Label)

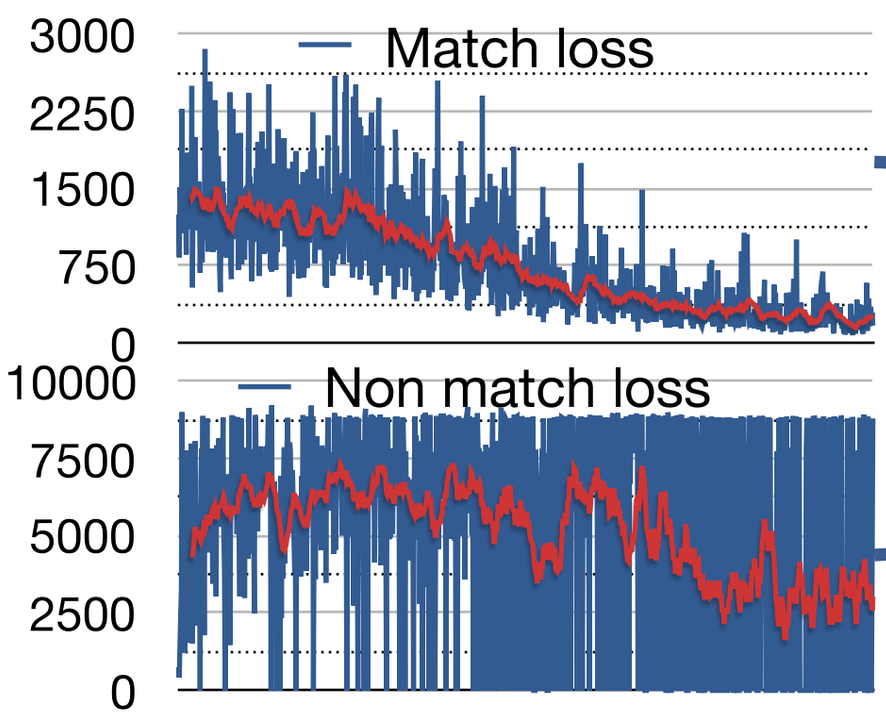


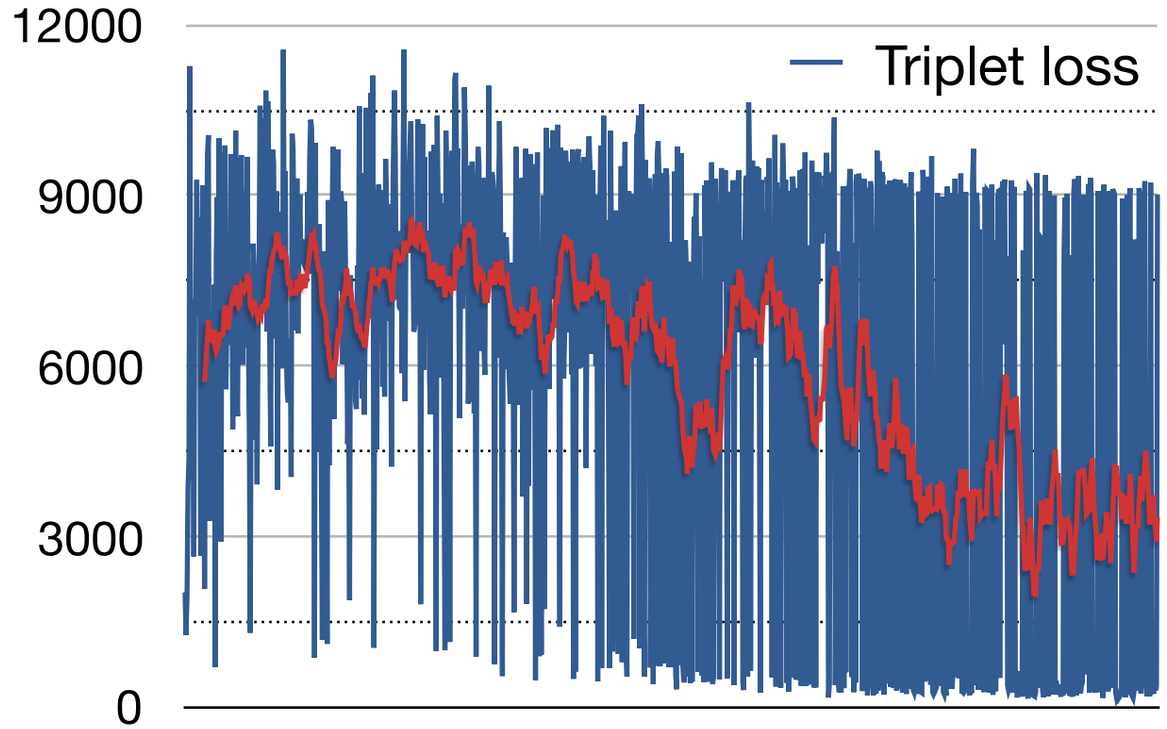
Training the Network

Trained on 14.000 TDF triplets | 32 triplets per iteration | 40 epochs

- ➡ Noisy Loss
  - ➡ Partly due to relatively small batch size
  - ➡ NonMatch loss = 0 if all above margin
- ➡ Distance means gradually move apart
- ➡ Margin controls ratio of losses

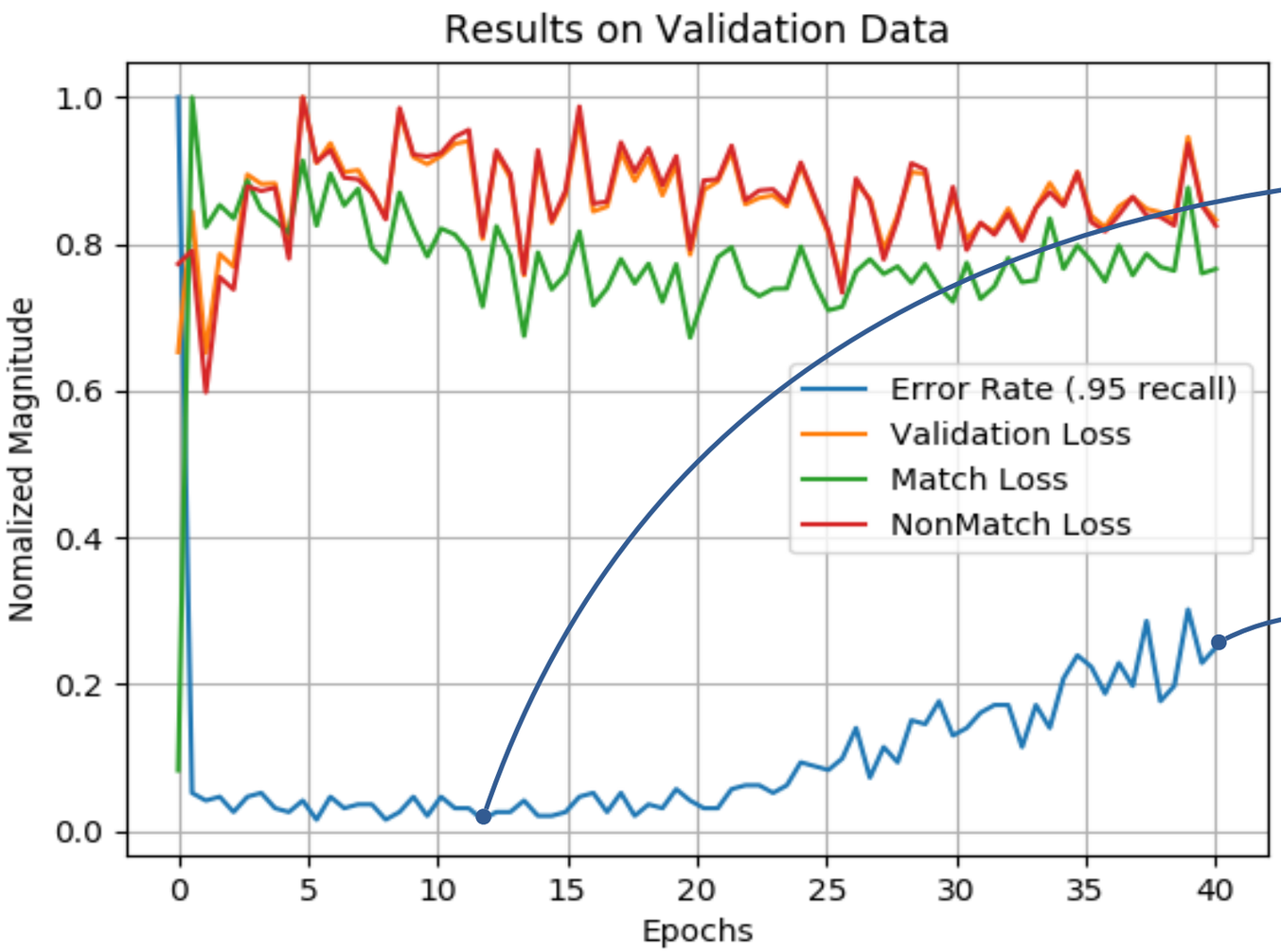


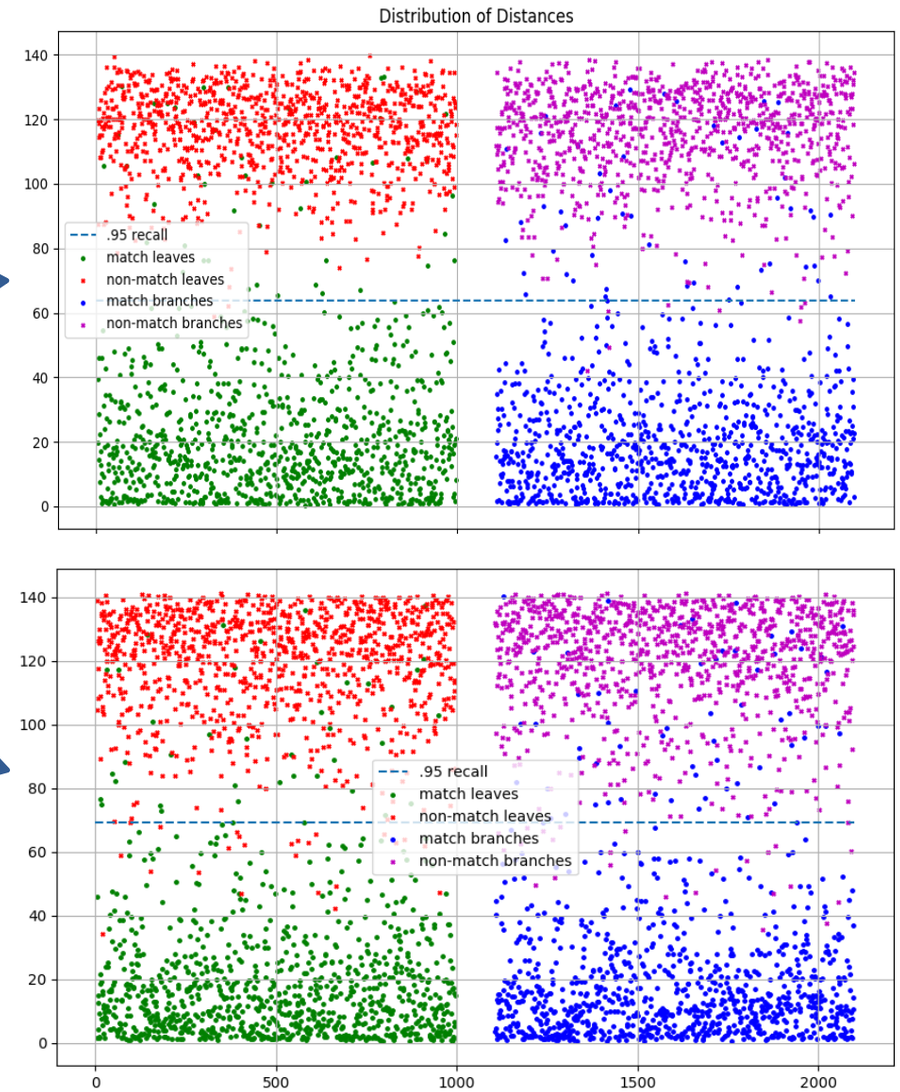




Validating the Network

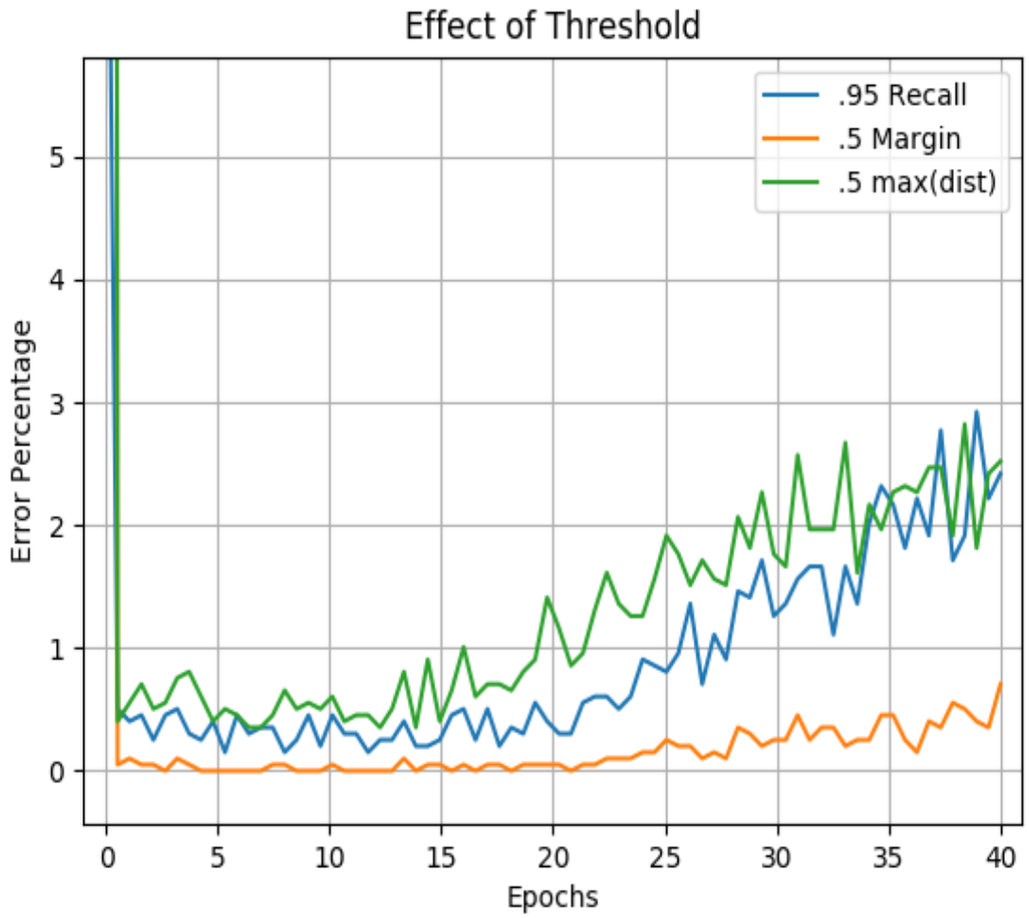
- The validation error quickly drops below 0.5% within the first few epochs
- Error starts diverging after ~20 epochs
- No corresponding increase in the validation loss



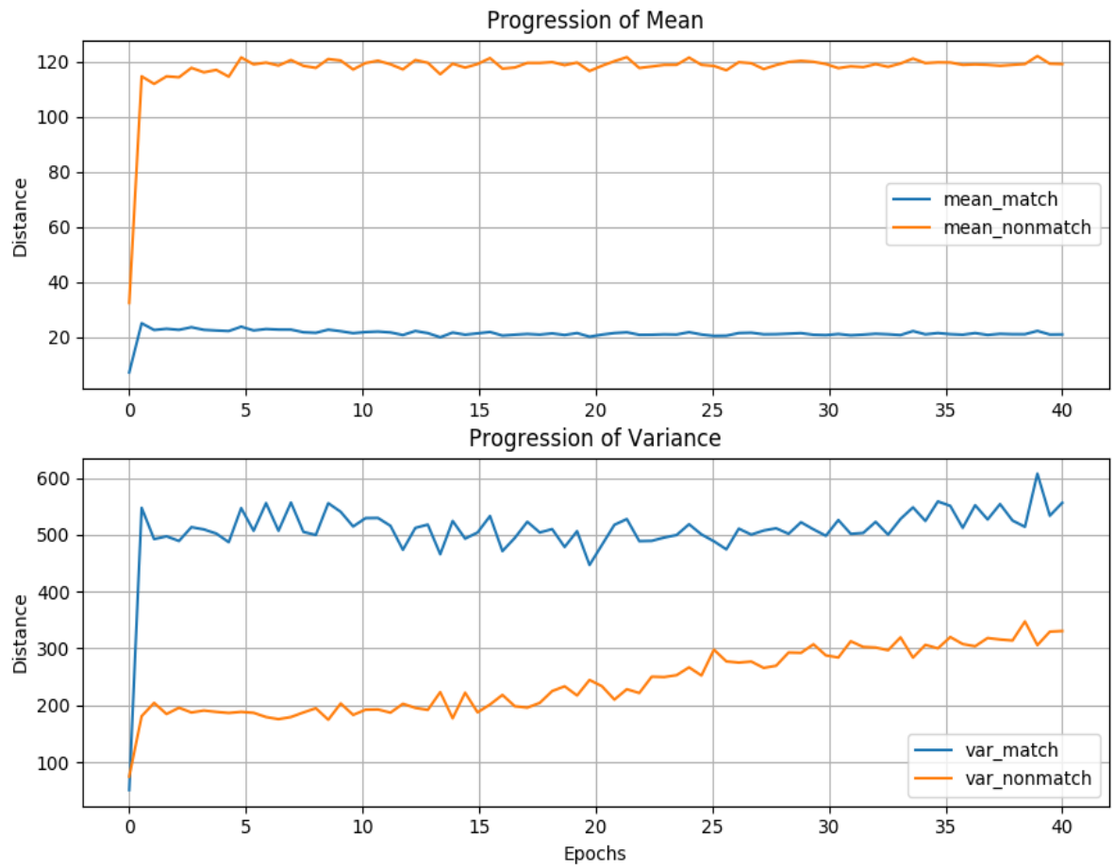


Validation set: 2.000 matches, 2.000 non matches | Batch size 32 | Shape of one TDF (30, 30, 30)

Analysis of Validation



Choice of threshold/ decision rule has only limited effect on error divergence

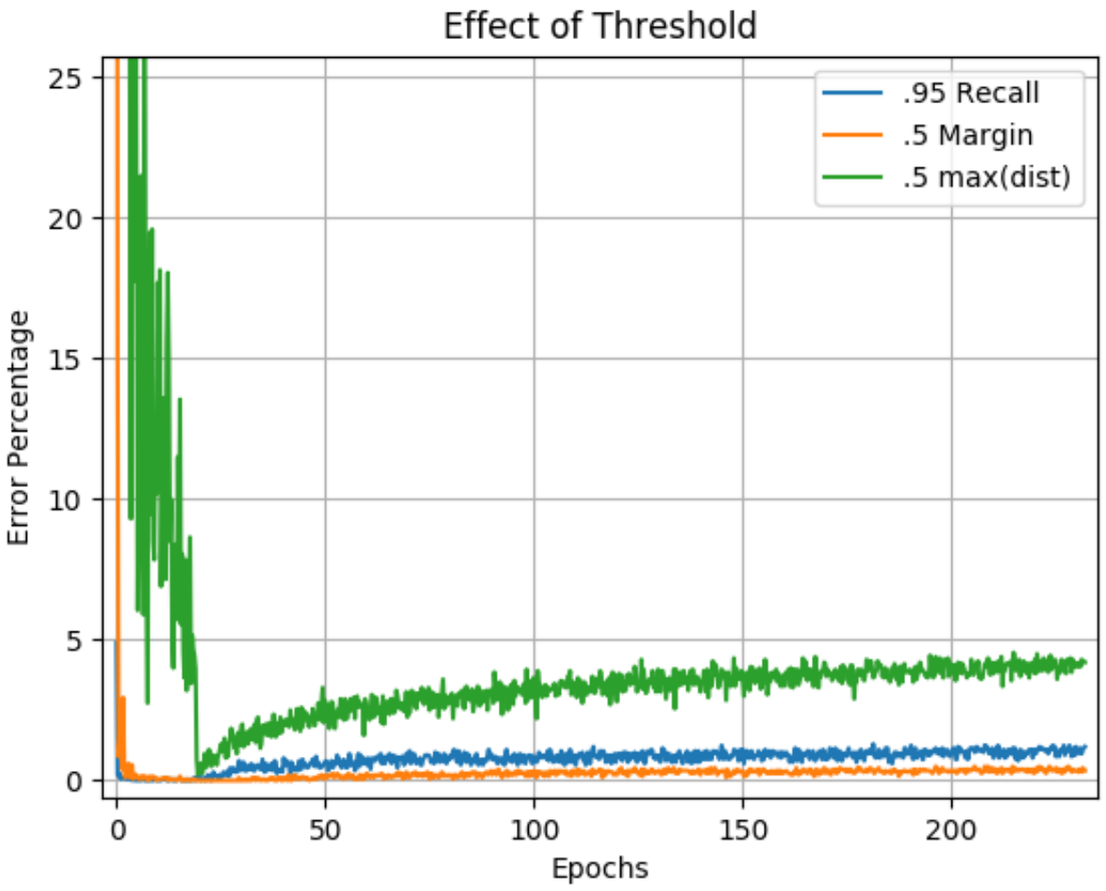


Error divergence is due to an increase of variance in the distance distribution

The loss is optimising the respective mean while an increase in variance is not punished & an increasing number of outliers affects the error rate

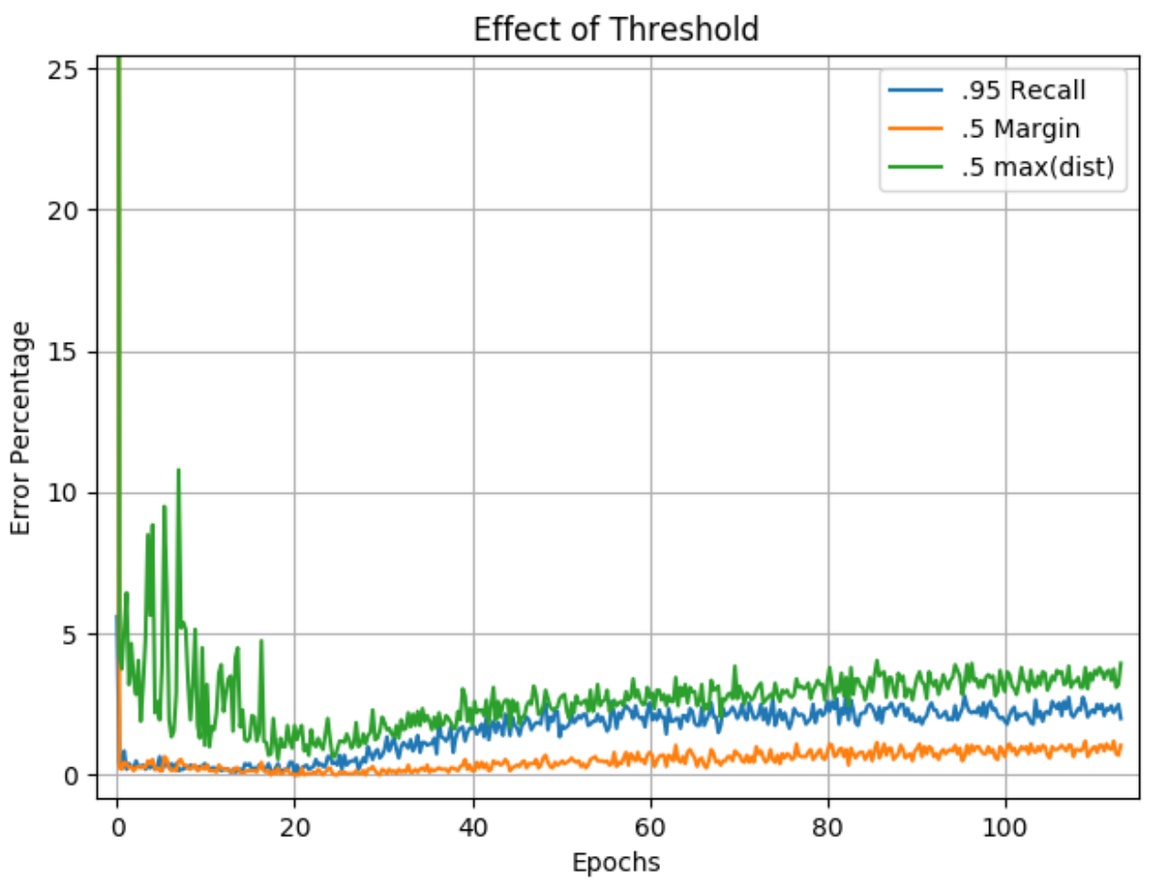
Further Analysis

Penalised Variance



With penalty on the sum of variances

Bias towards ‘hard samples’

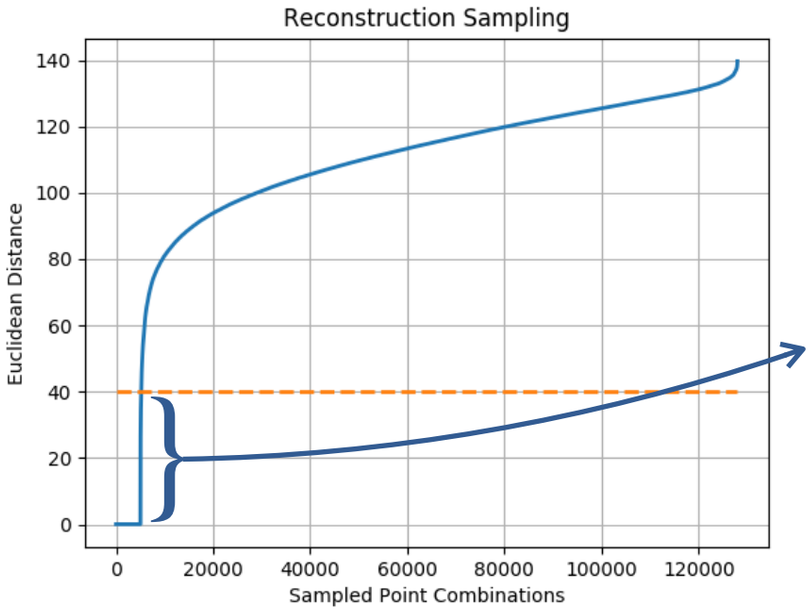


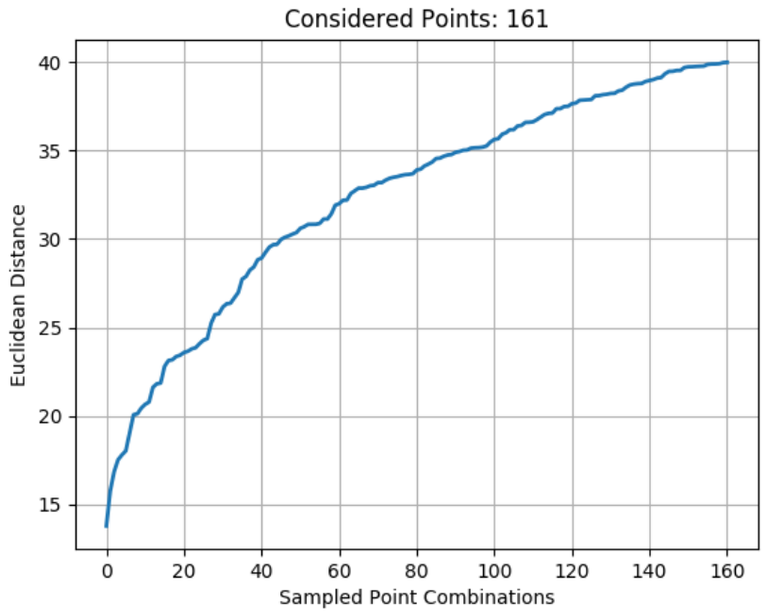
‘Hardest’ 25% of samples used in loss

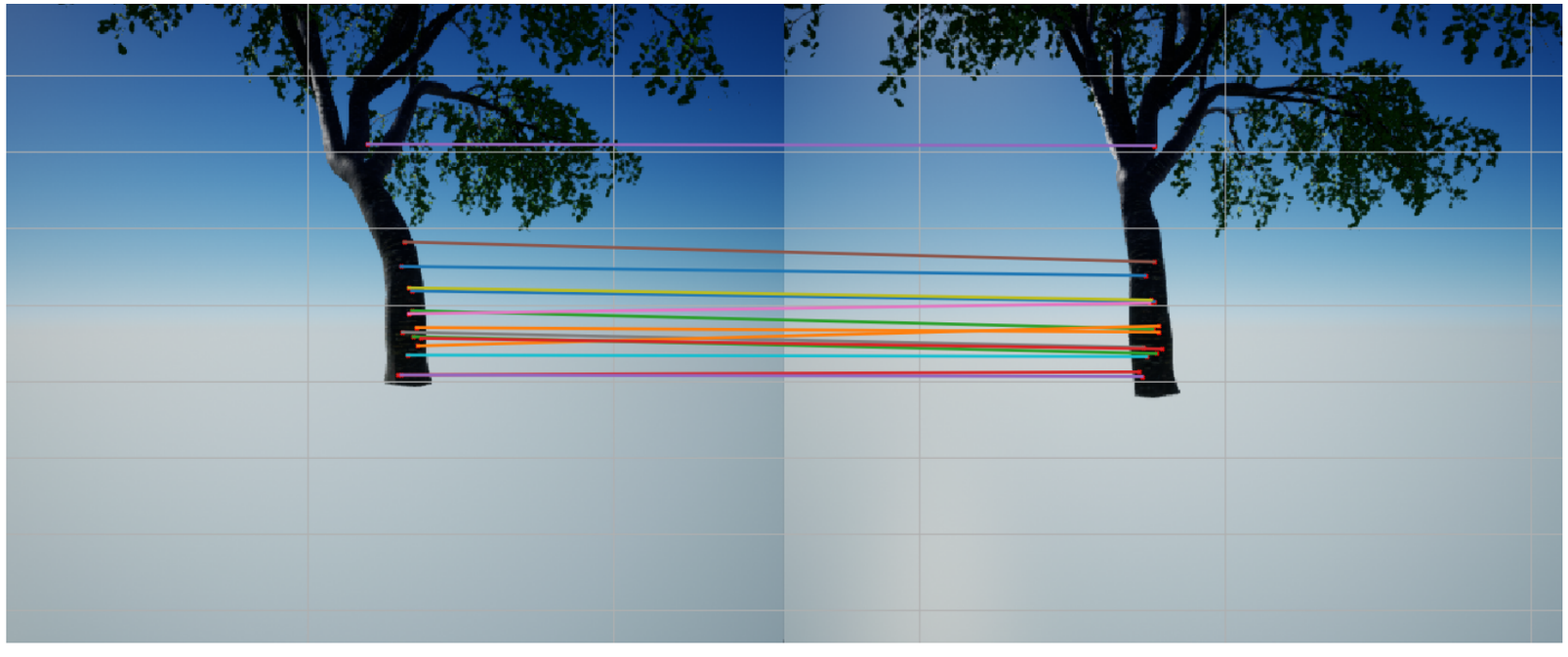
The effect of error rate divergence could be reduced but the overall characteristics of the validation performance remains the same

Reconstruction

Reconstruction







Sample grid of TDFs for each frame

↓

Feed random TDF combinations into the siamese network

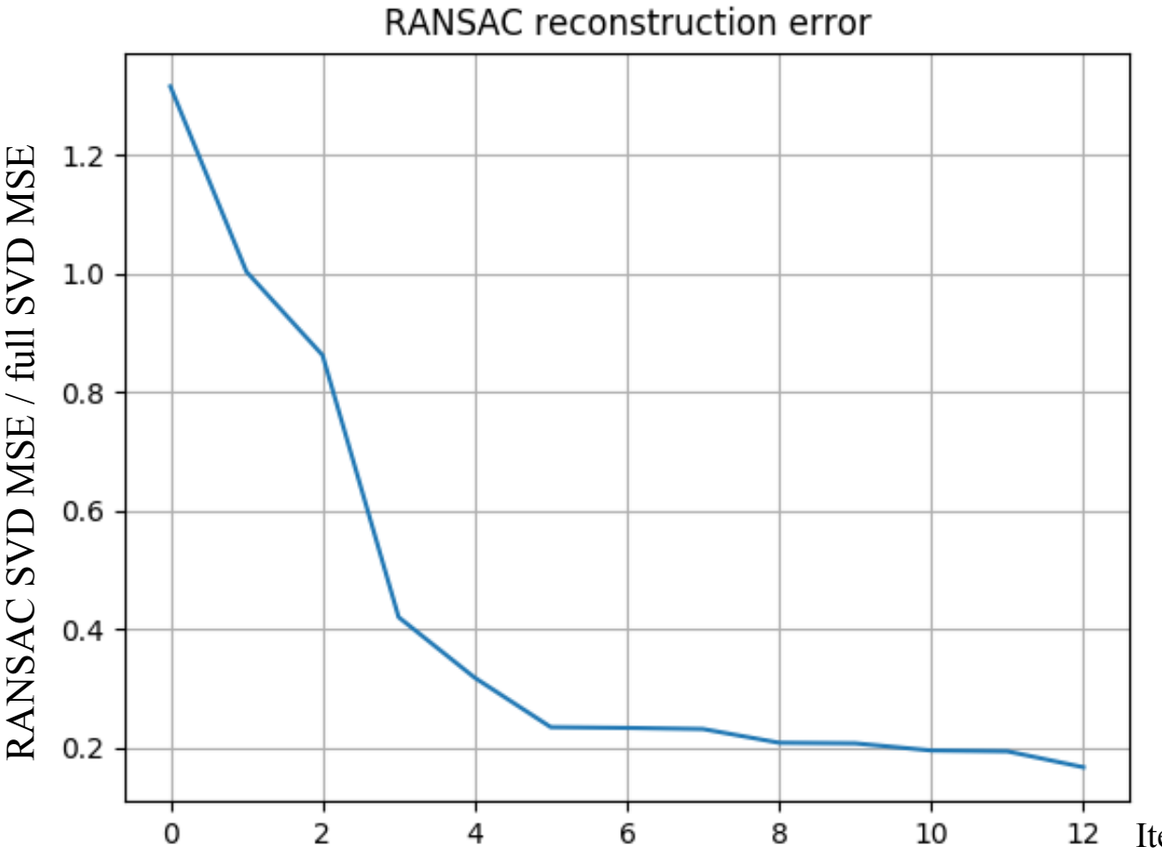
↓

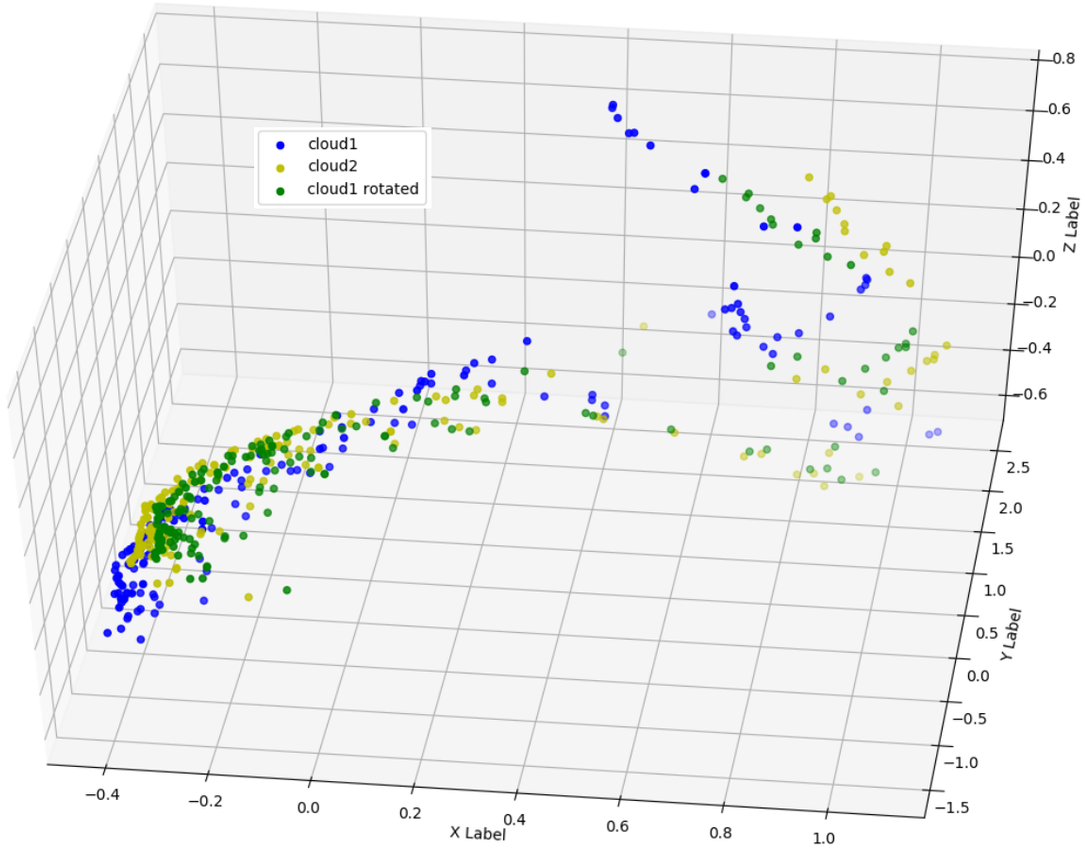
Choose combinations with lowest distance in embedding space

↓

Set of predicted correspondences for given frame pair

Reconstruction





Reconstruction of 2 correspondence point clouds using RANSAC + SVD (rmse: 0.652, strong correlation to decision rule)

- ➡ Reconstruction of correspondences using RANSAC + SVD approach
- ➡ Still remaining error in the rotation (Green should align with yellow)

Remaining and future work:

- ➡ Use and compare to more advanced reconstruction approaches
- ➡ Enforce wider distribution of correspondences for more stable reconstruction

Conclusion

Problem

- ➡ Generate training data from RGBD images
- ➡ Build siamese convolutional network
  - ➡ Train, optimise & analyse performance
- ➡ Predict correspondences in image sequence
- ➡ Use correspondences for reconstruction

Limitations

- ➡ Trained & evaluated on synthetic data
  - ➡ However provides a good starting point for future work on real data
- ➡ No benchmark comparison available

Results

- ➡ Data generation pipeline from RGBD image- and pose sequences to TDF triplets
- ➡ Quick convergence of error below 0.2%
- ➡ Early error divergence due to variance increase
- ➡ RANSAC reconstruction

Outlook

- ➡ Optimise loss function
  - ➡ Penalise/ reduce distribution variance
- ➡ Use real tree data sets (once available)
- ➡ Improve reconstruction error
- ➡ Compare validation- & reconstruction-error to benchmark approaches

Questions?

Appendix

Defining Performance

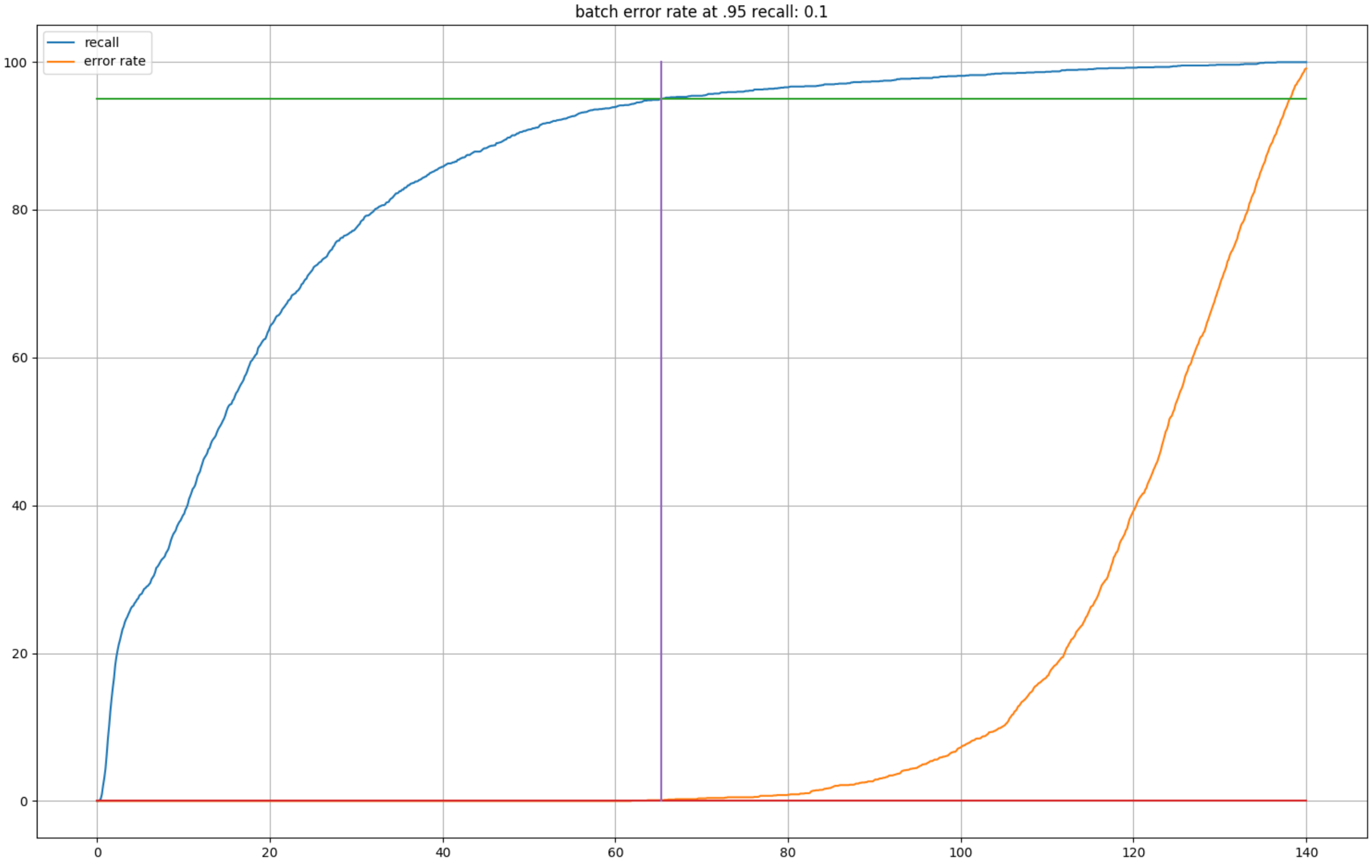
Truth \ Prediction	Match	NonM.
Match	**TP**	**FN**
NonM.	**FP**	**TN**

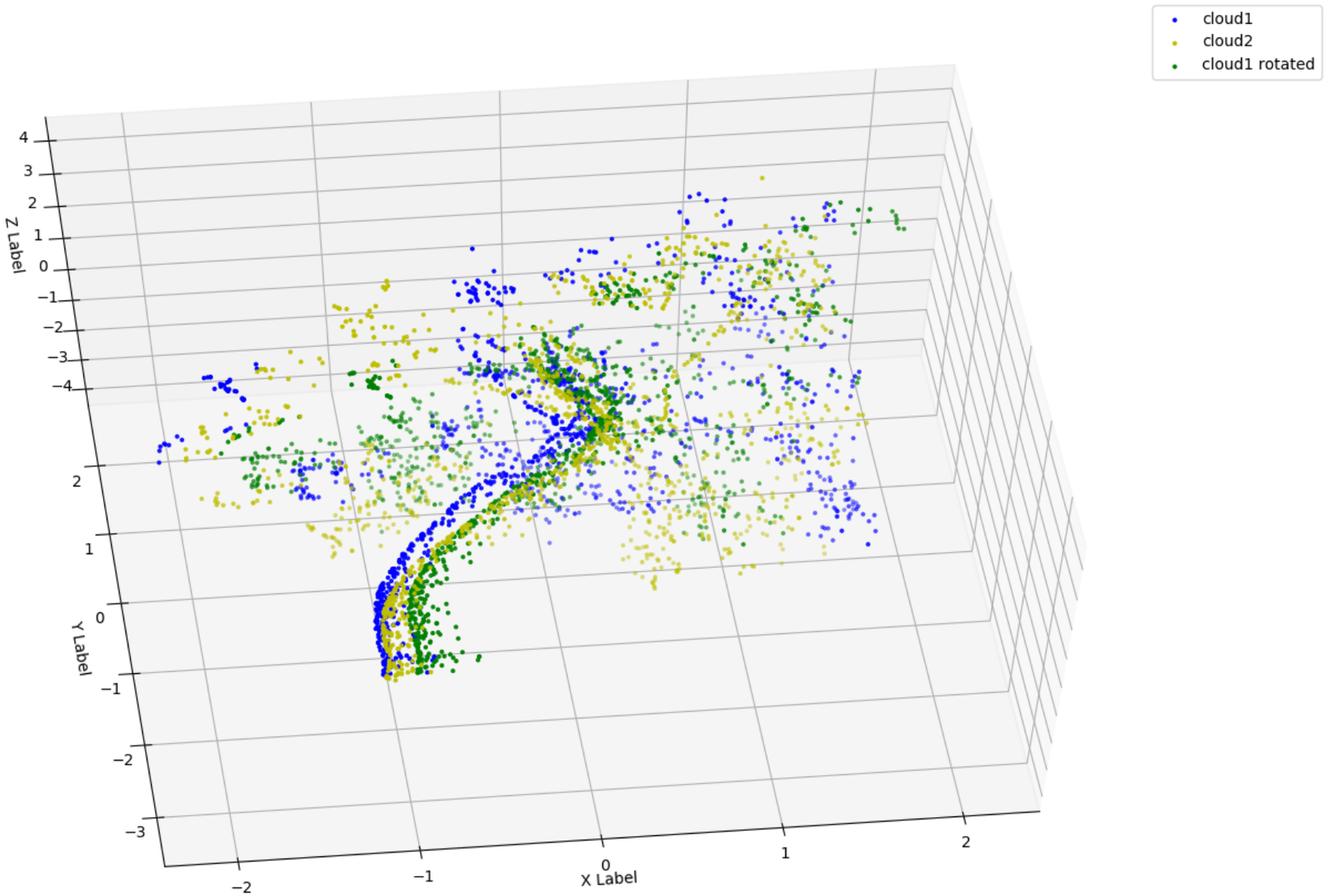
$$Recall = \frac{\#\ True\ Positives}{\#\ True\ Positives\ +\ \#\ False\ Positives}$$

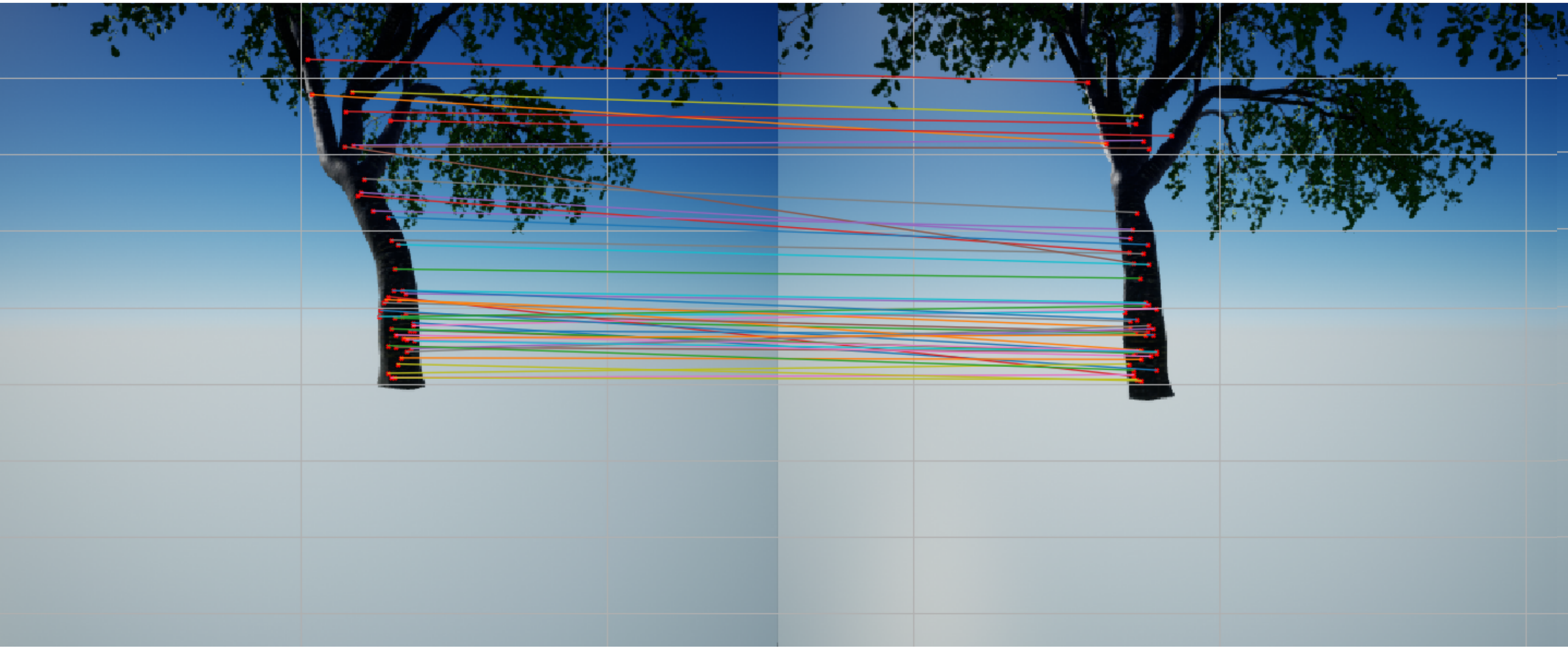
What proportion of the actual positives was identified correctly?

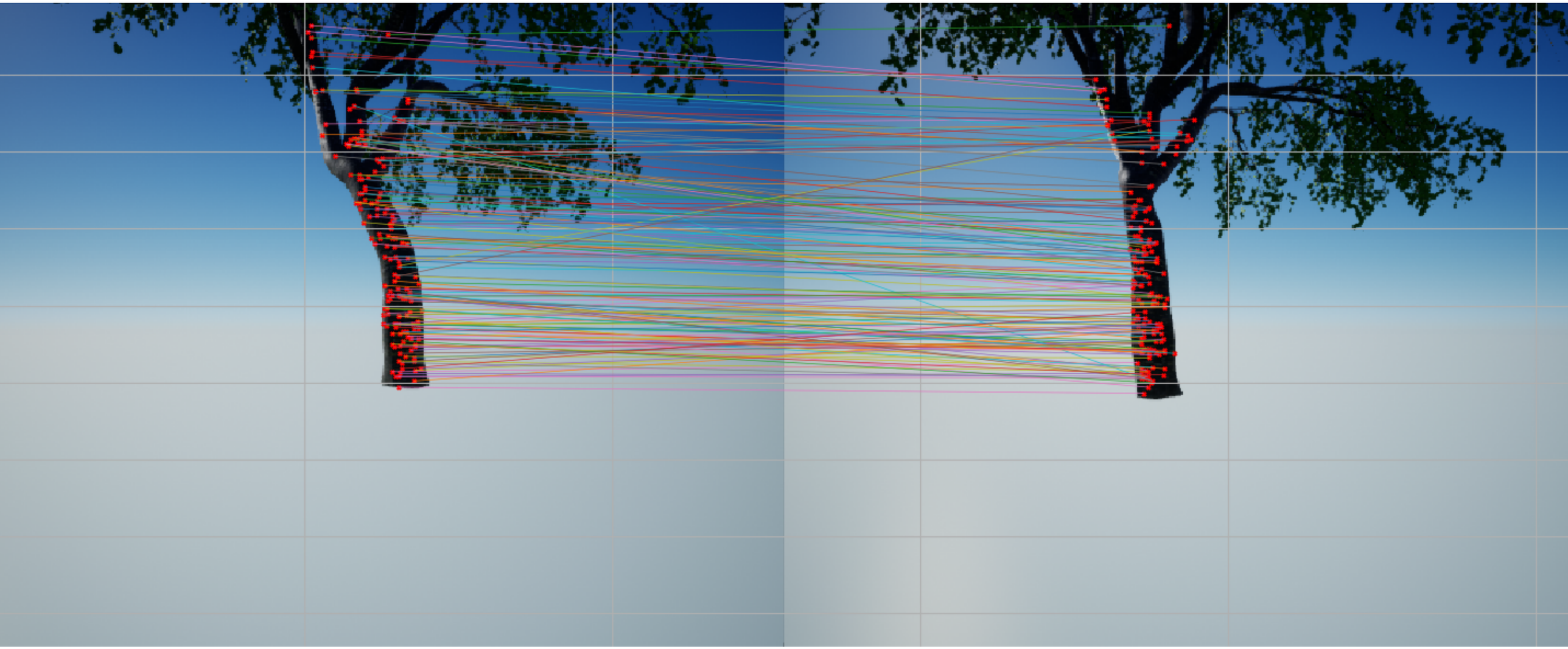
$$Error\ = \frac{\#\ False\ Positives}{\#\ False\ Positives\ +\ \#\ True\ Negative} = \frac{\#\ False\ Positives}{Constant}$$

What proportion of the predicted correspondences is incorrect?

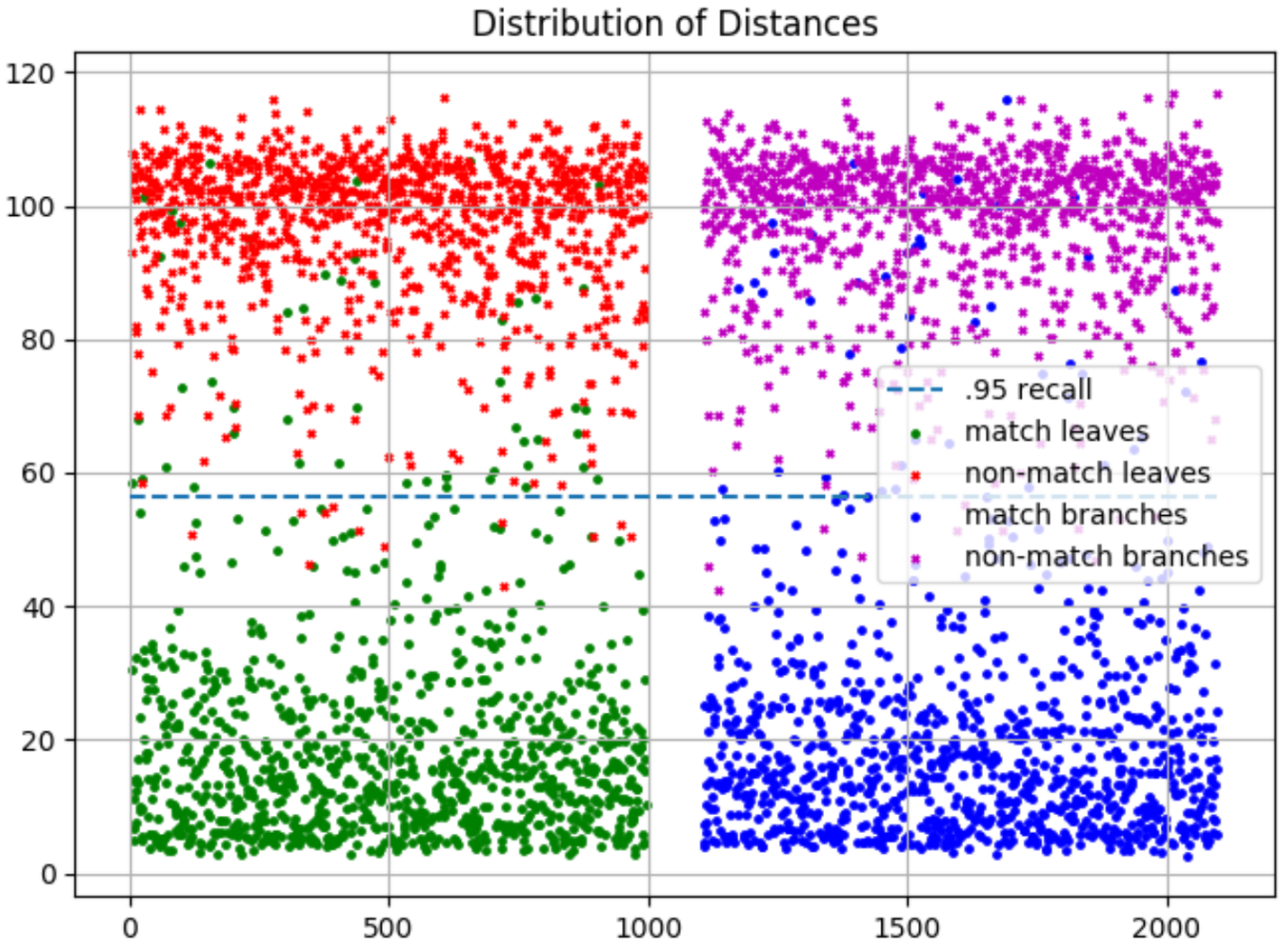

batch error rate at .95 recall: 0.1
recall
error rate
100
80
60
40
20
0
0
20
40
60
80
100
120
140

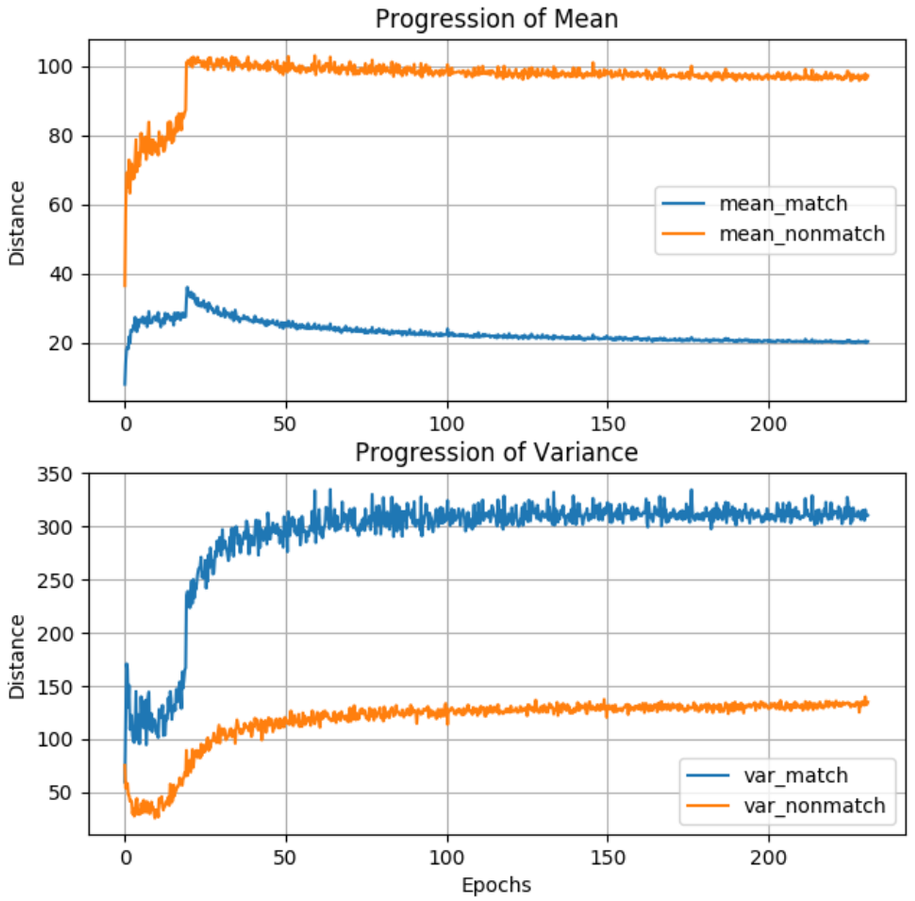

cloud1
cloud2
cloud1 rotated
Z Label
4
3
2
1
0
−1
−2
−3
−4
Y Label
2
1
0
−1
−2
−3
X Label
−2
−1
0
1
2

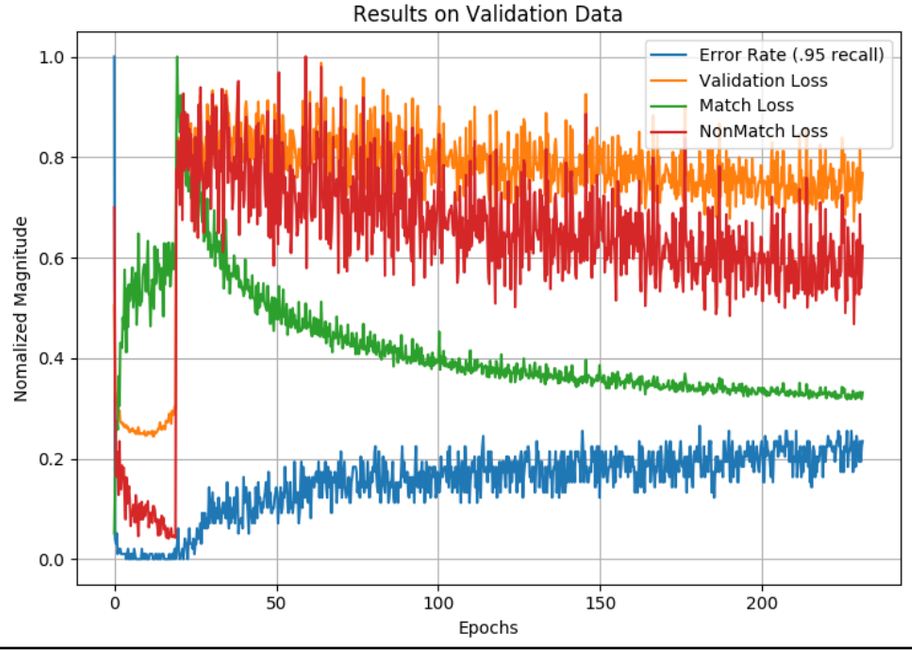




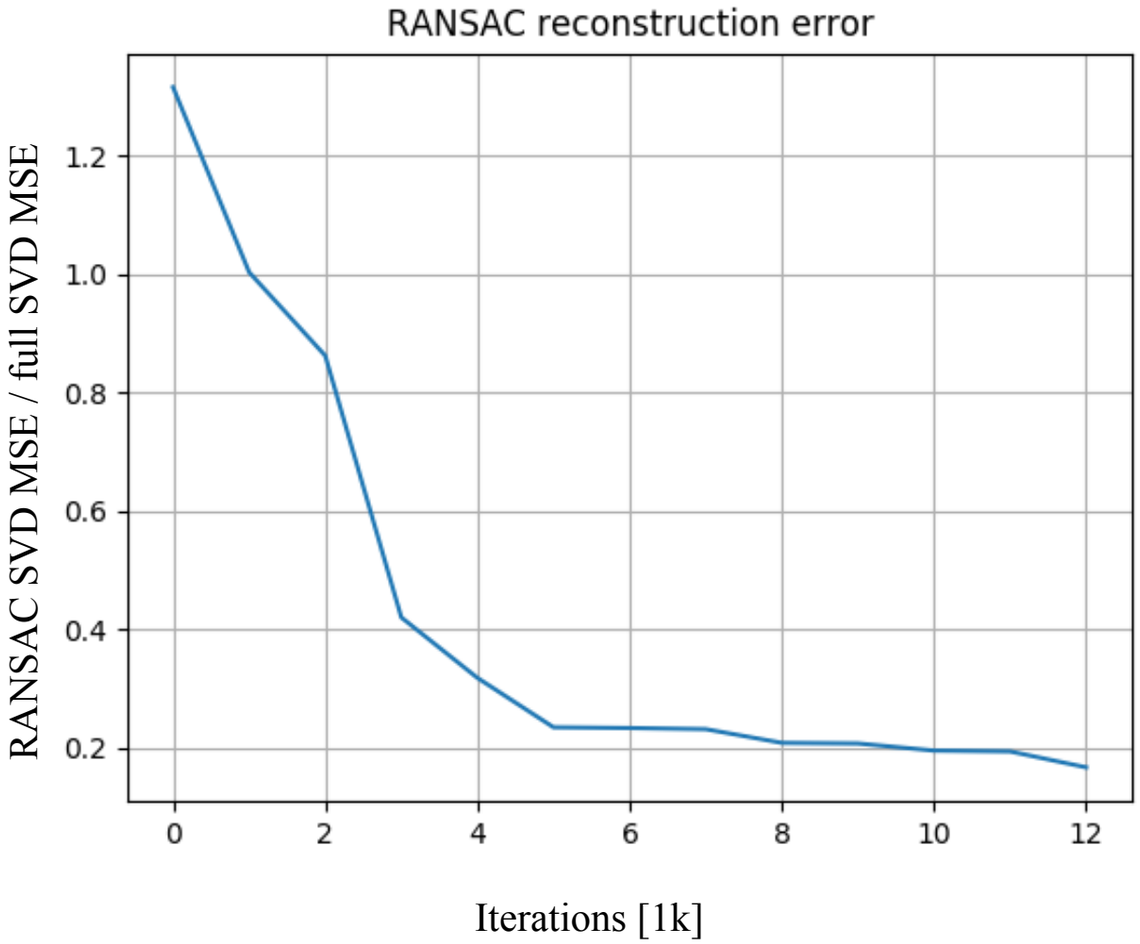
Variance Penalty Results



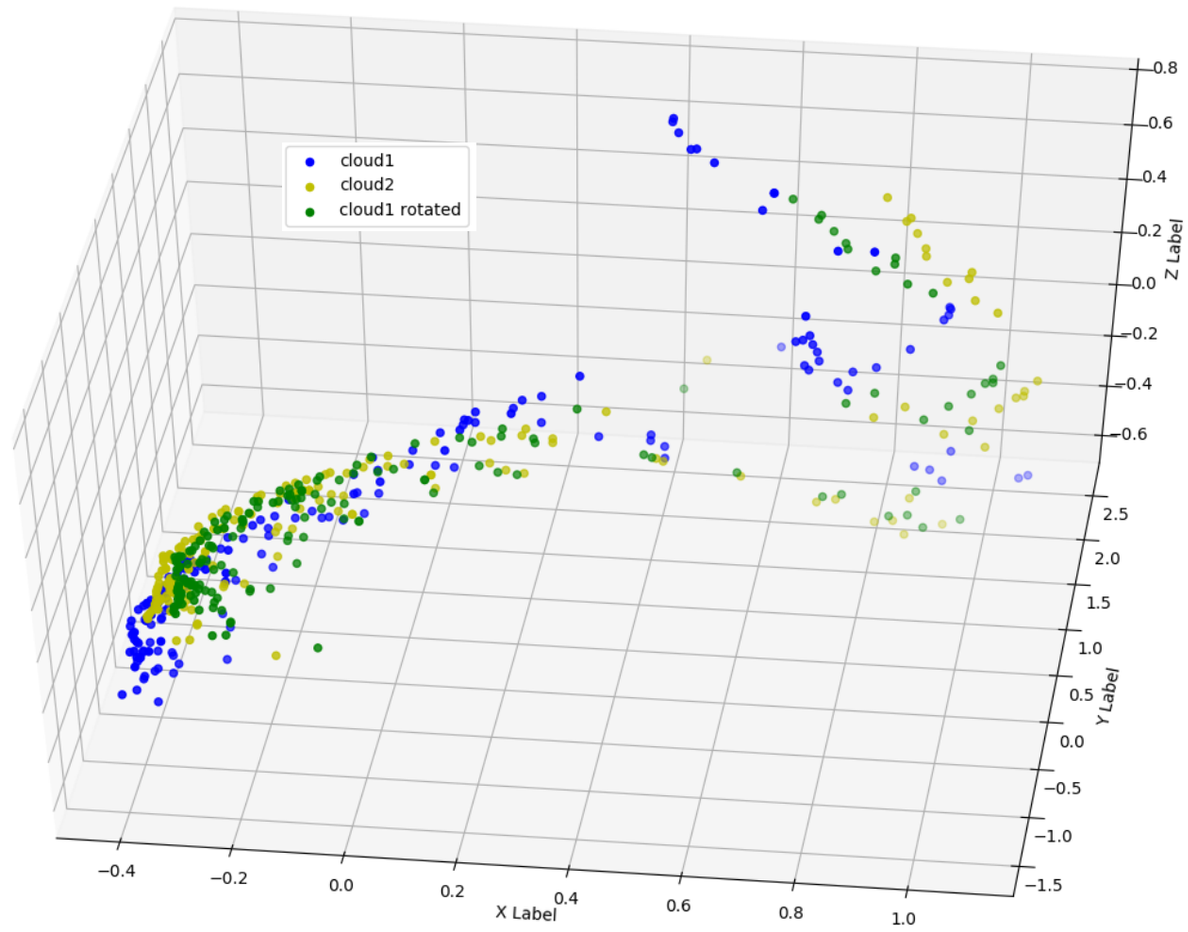




Reconstruction



- ➡ Error is optimised using RANSAC + SVD
- ➡ Might be reduced with other approaches
  - ➡ Pose estimation algorithms
  - ➡ Biased training data



- ➡ 2 correspondence point clouds (B,G)
- ➡ Respective rotation and translation (Y)

  Green and yellow should align

  Mean squared error: __

Contrastive / Triplet Loss

Intuition: Push means of match- and non-match-distances in the embedding apart

$$L_C = (Y - 1)\ D_W^2 + Y\ \{\max(0,\ m - D_W)\}^2$$

L_C - Contrastive Loss | D_W - Euclidean Distance | Y - Label | m - Margin

Alternative implementation: **Triplet Loss** (3 TDFs instead of 2 TDFs + Label)

Additional Processing:

I'll read 3dmatch and the other paper again to correct/ optimise this part

With the computed embedding distance, the 'difficulty' of the samples used for back propagation can be controlled. Either with a constant bias towards harder examples or dynamically to 'steer' the learning process from easy/ more general gradually towards harder samples

Pro: More control over learning process and possibly faster convergence

Con: Memory requirements. Only possible to some extent on systems of considerable size