A Combined Neural and Temporal Approach for Tracking Anatomical Features in Liver

Mélanie Bernhardt
ETH Zurich
melanibe@ethz.ch

***Abstract*—Ultrasound motion tracking is required for various medical applications. In this report, we describe a combined approach for tracking anatomical landmarks in liver during respiration, based on the CLUST Challenge. The proposed method combines a local Siamese-CNN and a Ridge Regression temporal model for feature localization at each frame. The method was developed and fine-tuned via 5-fold across-sequence cross-validation and then evaluated on the CLUST Challenge Test set.**

I. Introduction

Ultrasound (US) is a non-invasive, low cost, widespread medical imaging technology. It provides high temporal resolution measurement and is thus well suite for tracking of anatomical landmarks over time. Indeed, various medical applications require tissue motion tracking. Some examples of these applications are surgery simulation and training, neuroscience or any image-guided therapy procedure. Hence, it is of prime interest to develop accurate algorithms allowing precise tracking over time. In this project, based on the CLUST Challenge [1], a combined CNN-temporal approach is proposed for anatomical feature tracking in the liver during respiratory movements.

A. Related Work

1) Convolutional Network for Object Detection: Convolutional Networks have been heavily used in various computer vision tasks in the past years. Numerous examples where CNN were applied in the field of medical imaging can be found in the literature such as for example in MRI brain image analysis as in [2] or in radiology image analysis, see [3] for a review. On the other hand, neural networks have also been widely used for object detection. A review of deep learning approaches in this context can be found in [4]. However, for this particular task, we aim to track only one feature at a time but several features can be found on the input image making it challenging for the CNN to detect the right feature to track.

A Siamese Convolutional Network approach for the CLUST Challenge was recently proposed by [5]. In this work, the CNN learns the similarity between pairs of image regions containing the landmark. The network employs a temporal location prior to prevent the network from falsely switching from one feature to another.

2) Combined global and local approaches for improved tracking: [6] presented a 2-step procedure for global and local displacement tracking. First a block-matching algorithm finds a template containing the feature at the current frame. Then a localized block matching procedure using multiple templates positioned at the estimated contours of the features is used to refine the first estimate of the feature localization. The work from [6] currently obtains the best score on the CLUST Challenge with a mean tracking distance error of 0.72 mm tracking error on the test set.

B. Proposed approach

In this work, the goal is to combine ideas from the current state-of-art algorithms tackling the CLUST Challenge. It uses Siamese Neural Network as in [5]. However, the method proposed here differs from [5] in several ways. First of all, in [5] the network operated on a big fixed search region for all the frames. In this project, the network performs a local search on a small template whose location is dependent on the previous prediction, similarly to [6]. Moreover, as in [5], it was observed that without some temporal model, the network was jumping from one feature to another over time. Whereas the temporal regularization was introduced as a prior in [5], in this approach the CNN and the temporal model are treated as two separate models, proposing two independent predictions for the feature location. Finally, an additional step is introduced in order to detect and correct for absurd predictions when the network looses track of the feature over time.

II. Methods

A. Overview of the procedure

For each frame, two separate feature location search are performed: one is based on a temporal model taking into account the 5 last locations of the feature whereas the other is a Siamese Convolutional Network operating on a subset of the image extracted around the last known location of the feature. The temporal model is presented in section II-C whereas the CNN model (referred to as "Local Net") is presented in II-B. Each model predicts the location of the feature independently from the other model. If the predictions differs by more than a certain threshold in terms of Euclidean Distance, the final prediction is the average between them. Fig. 1 summarizes the procedure. However, as mentioned earlier, during the development of the method it was noticed that the Network occasionally jumped from one feature to another during tracking. This lead to an extremely bad performance of the tracking of the concerned feature as the network usually kept following the "wrong" feature until the end of the sequence. In order, to

avoid this kind of phenomenon, an additional safeguard test was added before returning the center prediction. Based on investigation on the annotated data at hand, it was noted that it never occurs that the difference between the center of the feature and the initial location is above 30 pixels for more than 50 frames. Hence, if it is detected that the prediction is lying in this absurd range for more than 50 frames, the prediction is reset to the initial location of the feature. Indeed, such a situation only occurs if the network has completely lost the initial feature to track or jumped to another feature to track. Resetting the position to the initial position is therefore the best guess possible in the hope that this will help the network will help the network to find feature again in the next frames. The pseudo-code corresponding the full prediction procedure can be found in the Appendix.

As a side note, initially, an approach similar to what was done in [6] was implemented. In this approach, the first step provided a coarse estimate of the displacement of the features by a vanilla block matching procedure aiming to extract a 60x60 template from the current frame based on the highest normalized cross-correlation with the 60x60 template extracted from the preceding template. After the global estimation of the feature displacement from frame to frame with the block matching algorithm, a second feature search using the Local Net was performed in order to refine the estimate of its location. However, the results on the cross-validation proved that the first block-matching step was superfluous. Indeed, simply using the feature location obtained at the preceding frame as the center of the search template for the current frame yielded results that were just as good as the results obtained via the two-step procedure. Hence, the simpler (and faster) method was preferred and the block matching step was abandoned.

Finally, the last procedure that was investigated in this project consisted of training two local features CNNs: one localizing the feature on a 60x60 pixel template and the other on a 20x20 pixel template. Both prediction were then averaged and used as the network prediction. The rest of the procedure stayed the same. However, as it will be shown in the results section, this double prediction did not improve the performance of the tracking.

B. Convolutional Network Feature Localization

1) Local Net training set: For each sequence in the training set from the CLUST Challenge, feature location annotations were available for a subset of frames in the sequence allowing to train our Local Net.

The training set for the Local Net was constructed as follows: for each sequence, for each feature, the reference template centered at the true feature location was extracted from the first frame. Then, for each annotated image, a Gaussian Noise $\mathcal{N}(0, 5)$ was added to the true center location of the feature. A template centered around this perturbed center was then extracted from each image. The goal of the network was then to learn the coordinates of the true center given the input triple (perturbed template, reference template, perturbed center) for each image in the training set. Any preceding frame could be as a reference template to augment to training set i.e. not only the triple (perturbed template, first template, perturbed center) were considered but also any other triple (perturbed template, previous template, perturbed center). Finally new random perturbations and their corresponding templates were computed at each epoch in order to increase the training data. An example of such template pairs is shown on Fig. 1.

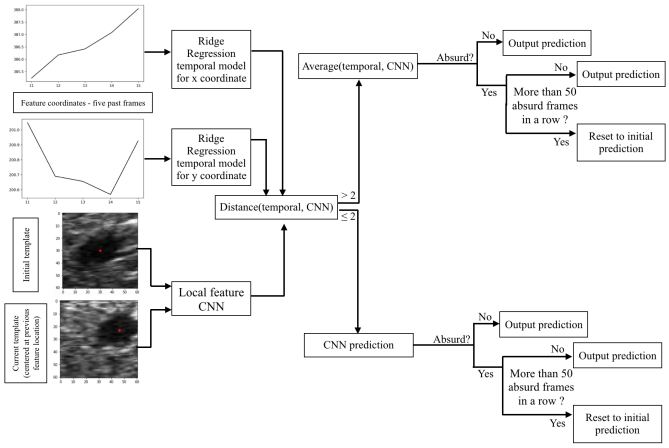

Fig. 1: Overview of the hybrid prediction procedure

2) Local Net Architecture: A schema of the Local Net architecture can be found in Fig. 2. Both templates are processed by the same convolutional network (i.e. weight sharing during training). The convolutional network is composed of 2 blocks of 2 convolutional layers. Between each block a dropout layer was added to reduce overfitting. Batch normalization was used between each convolutional layer. After the convolutional part of the network a dense layer is used to produce a 256-unit embedding of each template. These embeddings are then concatenated and feed into a 128-unit fully connected layer followed by a 2-unit fully connected layer. The output of this last layer corresponds to the relative coordinates of the feature location with respect to the center of the input template. In order to output the coordinates of the feature location in the original (full) image, the center coordinates of the input template is added to these local coordinates. Note that several standard architecture were tested throughout the project (e.g. AxelNet, VGG-16), however these architecture performed worse in the cross-validation evaluation that the simpler architecture described above. The loss function optimized during training was the pixel-wise euclidean distance. The network was trained with the Adam optimizer with an initial learning rate of 0.001 for 15 epochs (30,000 training steps).

C. Temporal Feature Localization

The temporal model employed is a simple Ridge Regression using the five last feature location as predictors. As the training set does not contain annotation for each frame in the sequence, a linear interpolation was used to compute the missing annotations on the training set (similarly to [7]). Several temporal

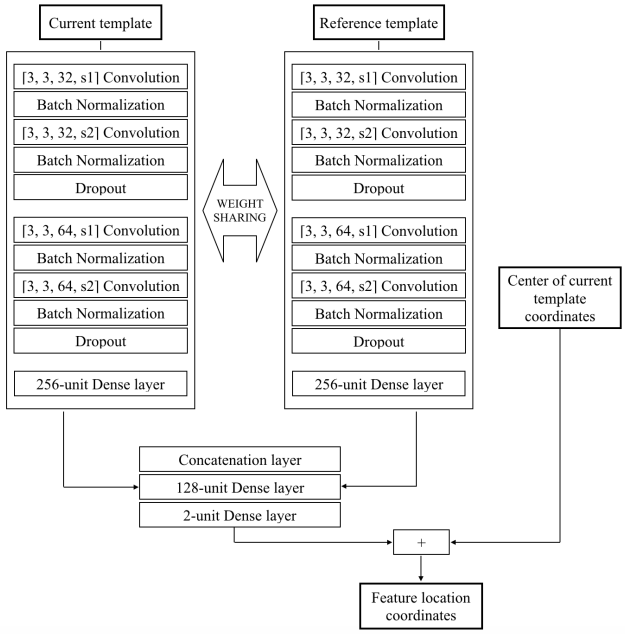

Fig. 2: Architecture of LocalNet. Note that the notation [3, 3, 32, s1] means convolution with filter size [3 x 3], 32 channels and a stride of 1. The dropout rate was set the 0.5.

features, window sizes and models were tested via a cross-validation on these interpolated datasets. Despite its simplicity, the best model found was a Ridge Regression with the 5 last observed features location as predictors. The cross validation procedure used to find the best temporal model was based on the assumption that the interpolated annotations used as training values are correct. These results do not show the quality of the model to estimate the motion on the long term but merely its quality given the true previous locations. However, at test time the true previous location will not known and merely the five last predictions can be used as predictors for the next frame. Hence, if the network failed in the 5 last predictions, especially at the beginning of the sequence, the temporal model will also fail to locate the feature as relies on the quality of the 5 last predictions. Due to the linear interpolation of the annotations at training time, the model mainly learns a simple linear regression model based on the five last predictions. Using solely the temporal model would lead to a roughly linear prediction that would not capture the variations in the feature location on the long run. However, the cross-validation evaluation presented in the next section, show that this extremely simple model can still help to improve the overall performance of the motion tracking.

III. Results

A. Evaluation procedure

In order to assess the performance of our method, an "across-sequence" 5-fold cross validation procedure was implemented. In other words, for each iteration in the cross-validation, the network was trained on 20 of the training sequences (i.e. approximately 80 training features) and validated on the remaining 4 sequences (approximately 10 testing features per fold). Several template sizes were assessed via this cross-validation setting. However, only the experiments with a template size of 60 x 60 will be presented here, as this size was found to be the optimal template size for the proposed algorithm. Several thresholds for the number of "absurd prediction" limit were tested. As explained in the last section, a threshold of 50 means that if the predictions were in the absurd region (i.e. the region were the difference between the predicted coordinate and the initial coordinate is above 30) for more than 50 frames in a row, the prediction is reset to the original feature coordinates.

B. Experiments results

Table I presents the results obtained via the above-mentioned across-sequence cross-validation. The experiments shows the impact of the threshold choices for the absurd prediction limit as well as the use of the temporal model (cf. schema 1). We can see in Table I, that the use of the "safeguard" procedure i.e. resetting the prediction when the network has lost track of the feature has a big impact on the average performance of the method. Indeed, without this step the average distance between the predictions and the true locations is doubled (all other parameters equal). Secondly, it is interesting to note that always using the temporal model decreases the performance of the procedure compared to the model that does not use the temporal model at all. This is due to the fact that the temporal model is very simple, hence it only provides a rough estimate of the location of the feature. However, if we start using the temporal model only only if the network's prediction and the temporal prediction differ by more than 2 pixels, then the best performance is achieved. There is a simple heuristic behind this fact: if both prediction differ too much it indicates that the network probably fails to detect the feature correctly (jump in feature, uncertain prediction), hence taking the average prediction between the temporal model and the network prediction is a more robust estimate than either of the methods alone. Finally, the last line corresponds to the evaluation of the model using two CNNs simultaneously (operating on two different template size) as described in the last paragraph of Section II-A. Concluding from Table I, the best model found was the model using a CNN operating on a 60 x 60 template with a temporal model threshold of 2 pixels euclidean distance (i.e. using the temporal model only if the prediction from the network and the temporal model differ more than 2 pixels), with a absurd predictions threshold of 50 frames. For this model, the cross-validation mean euclidean distance was 1.31mm. In order, to compare the model presented in this project with the current state-of-the-art model in the CLUST Challenge, this model was used to predict the frames location on the test set and the results were submitted to the Challenge. The results obtained on the test set can found in Table II as well as in figure 3. For most of the features the tracking is very accurate (less than 1 mm). However, for 5 features of the test set the tracking feature

TABLE I: Across sequence cross-validation results. 60NetP30T5 denotes a CNN trained on a 60 x 60 template, with a absurd prediction threshold of 30 frames in a row, and a temporal model used only if the prediction from the network and the temporal model differ by more than 5 pixels. "NoSafeguard" means that the absurd predictions were not reset at all over time.

Model	Absurd prediction threshold	Temporal model threshold	Mean Euclidean distance (mm) across folds
60NetP30T5	30	5	1.82
60NetP40T5	40	5	1.51
60NetP50T5	50	5	1.48
60NetP50T2	50	2	**1.31**
60NetP50T0	50	0	1.70
60NetOnlyP50	50	-	1.51
60NetT2NoSafeguard	-	2	2.70
60Net30NetP50T2	50	2	1.60

TABLE II: CLUST test set results

Scanner	Mean Euclidean distance (mm)	Standard Deviation	95th Percentile
CIL	0,86	0,77	2,13
ETH	7,96	13,52	36,50
ICR	1,73	2,97	9,63
MED1	2,51	6,80	16,07
MED2	2,03	5,02	8,82
All sequences	5,32	11,04	26,73

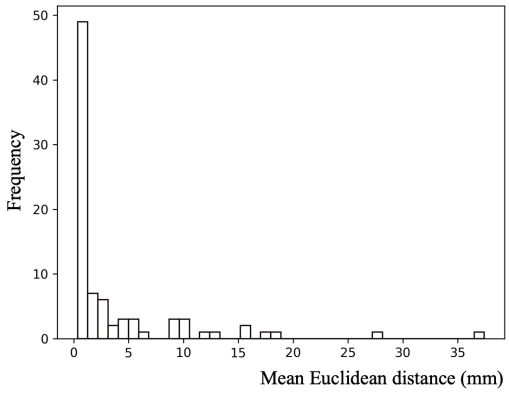

Fig. 3: Mean Euclidean Distance per test feature distribution (mm)

is extremely bad (with an average distance greater than 15 mm). These outliers lead to a significant drop in the average performance on the test set compared to the performance achieved in the cross-validation. Hence, the overall average tracking error on the test set is only 5.32 mm which is below the performances reported on the CLUST public leaderboard (best ten predictions reported online ranging between 0.72 and 3.34 mm).

IV. Conclusion

In this project, a method combining a Convolutional Network and a simple temporal model is proposed to track anatomical landmarks in liver during respiratory motion. The main advantage of the method is its simplicity and speed. It is very fast to train (less than one hour on GPU) and prediction are obtained extremely fast (approx. 100 frames per second) allowing real-time prediction. Similarly, the temporal model is trained in only a few seconds. Due to the simplicity of the method, it would be very easy to retrain the method over time when more data is acquired. Hence, this method could be convenient to use in a real setting where more and more data is acquired over time. Observing the tracking results video, shows that for 75 features out of 85 the method tracks accurately the feature throughout the complete sequence. However, it was observed that the method completely fails to detect some features for some particular scanner (approx. 8 features out of 85 test features). These outliers unfortunately have a high negative impact of the average performance of the method (cf. Fig. **??** showing the distribution of the errors) leading to an average euclidean distance of 5.32 mm on the test set which is lower than what was expected by the cross-validation tuning procedure. This issue could potentially be overcome by adding more training data from this type of scanner to improve the network performance. A possible improvement could also come from engineering a more complex temporal model introducing in order to detect when the network switches from one feature to another in a more principled way than what is done with the current 'absurd prediction zone' detection.

References

[1] V. De Luca, T. Benz, S. Kondo, L. König, D. Lübke, S. Rothluebbers, O. Somphone, S. Allaire, M. Lediju Bell, D. Chung, A. Cifor, C. Grozea, M. Günther, J. Jenne, T. Kipshagen, M. Kowarschik, N. Navab, J. Rühaak, J. Schwaab, and C. Tanner, "The 2014 liver ultrasound tracking benchmark," *Physics in medicine and biology*, vol. 60, pp. 5571–5599, 07 2015.

[2] J. Bernal, K. Kushibar, D. S. Asfaw, S. Valverde, A. Oliver, R. Martí, and X. Llado, "Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review," 12 2017.

[3] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug 2018. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9

[4] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *arXiv e-prints*, p. arXiv:1807.05511, Jul 2018.

[5] A. Gomariz, W. Li, E. Ozkan, C. Tanner, and O. Goksel, "Siamese Networks with Location Prior for Landmark Tracking in Liver Ultrasound Sequences," *arXiv e-prints*, p. arXiv:1901.08109, Jan 2019.

[6] A. J. Shepard, B. Wang, T. Foo, and B. Bednarz, "A block matching based approach with multiple simultaneous templates for the real-time 2d ultrasound tracking of liver vessels," *Medical Physics*, vol. 44, 09 2017.

[7] X. Chen, C. Tanner, O. Göksel, G. Székely, and V. De Luca, "Temporal prediction of respiratory motion using a trained ensemble of forecasting methods," in *Medical Imaging and Augmented Reality*, G. Zheng, H. Liao, P. Jannin, P. Cattin, and S.-L. Lee, Eds. Cham: Springer International Publishing, 2016, pp. 383–391.

APPENDIX

Algorithm 1 Combined prediction procedure

At the first frame initialize k to 0.
Start using the temporal model only after the 5 first frames.
procedure PREDICTCENTER(k, refTemplate, prevCenter, Image, FivePrevPredictions, initCenter)
    template ← EXTRACTTEMPLATE(Image, prevCenter)
    Center ← LOCALNET(template, refTemplate)
    TempX ← RIDGEX(FivePrevPredictions)
    TempY ← RIDGEY(FivePrevPredictions)
    TemporalCenter ← [TempX, TempY]
    if DIST(ImageCenter, TemporalCenter) $>$ 2 **then**
        Center ← MEAN(Center, TemporalCenter)
    if (prevCenterX - initCenterX)$>$30 OR (prevCenterY - initCenterX)$>$30 **then**
        $k \leftarrow k + 1$ ▷ Increase suspicious prediction counter
    else
        $k \leftarrow 0$ ▷ Reset k
    if $k > 50$ **then** ▷ Absurd case
        Center ← initCenter ▷ Reset the prediction
        Stop using temporal model for the remaining frames
    return Center, k