Supplementary Material

Somnath Dutta[1][0000-0003-3982-8780], Benjamin Russig[2][0009-0008-7724-7868], and Stefan Gumhold[2][0000-0003-2467-5734]

[1] CNR-ISTI, Pisa, Italy
[2] Technische Universität Dresden, Germany

1 Non-rigid Registration Pipeline

This section provides an overview of the non-rigid registration pipeline. Algorithm 1 outlines the key steps for deforming a plant point cloud. Further details are elaborated in the following sections, focusing on essential aspects.

Algorithm 1 Spatio-temporal registration

```
1: Input: $P_1, P_2$ {Input point clouds}
2: $S_1 \leftarrow$ ComputeL1MedialSkeleton($P_1$)
3: $S_2 \leftarrow$ ComputeL1MedialSkeleton($P_2$)
4: $(D_{\text{geo}}, D_{\text{ecd}}) \leftarrow$ ComputeDistances($S_1, S_2$)
5: $(T, E, \text{states}) \leftarrow$
   ComputeProbablisticRelationship($D_{\text{geo}}, D_{\text{ecd}}, S_1, S_2$)
6: $\tau \leftarrow$ ComputeCorrespondences($T, E, \text{states}$)
7: $T_f \leftarrow$ ComputeSkeletonDeformation($S_1, S_2, \tau$)
8: $\hat{P}_1 \leftarrow$ ApplyDeformation($P_1, T_f$)
9: Output: $\hat{P}_1$
```

1.1 Skeleton Extraction

The initial step involves extracting a skeleton S from an input point cloud P. Several methods for skeleton extraction are available as broadly discussed in our background research (**refer to main paper**). Chebrolu et al. employed a method that combined support vector machine-based segmentation with clustering to identify individual leaves or stems as distinct entities. Subsequently, they determined keypoints or skeleton nodes for each organ using self-organizing maps. However, our analysis observed no significant advantages in adopting methods that introduce additional overhead and the need for the segmentation approach of Chebrolu et al. compared to simpler but efficient techniques. Importantly, L1-medial skeletonization is a robust method to obtain skeletons of complex structures. Hence, we opted for the L1-medial skeletonization algorithm. Thanks to their adaptability to local point sets, L1-medial skeletons offer a robust representation of 3D point cloud data. Their construction algorithm effectively processes noisy, outlier-ridden, and incomplete raw scans, showcasing versatility across various shapes and structures. A typical plant point cloud skeleton from the dataset is represented by a significantly larger number of nodes, ranging from 75 to 120, compared to the method used by Chebrolu et al.. This higher node count can create computational bottlenecks for the Viterbi algorithm, whereas beam search is capable of efficiently handling graphs with a greater number of nodes.

1.2 Skeleton Matching

Hidden Markov Model: Hidden Markov Models (HMMs) emerged as essential tools for capturing the temporal evolution of plant structures. An HMM takes in a sequential series of observed states, state-to-state transition probabilities, and state-to-observation emission probabilities. The objective is to deduce the sequence of hidden states most likely to have generated these observations. The HMM model offers the flexibility to encode various cues, define constraints for correspondences, and incorporate prior information about the skeleton structure. This capability enables tracking multiple correspondence candidates and selecting the optimal correspondences between skeleton pairs. Adapting HMM into the spatio-temporal plant registration overlaps with the approach from Chebrolu et al..

For HMMs, the transition cost (T) and emission cost (E) need to be formulated. Transition cost refers to the penalty associated with transitioning between different states, reflecting the likelihood of moving from one state to another. On the other hand, emission cost represents the probability of emitting an observation (or symbol) from a particular state, providing insight into the likelihood of observing specific data given the underlying state.

The transition and emission costs are transformed to probabilities using a simple function $p(x) = \frac{1}{x}$, where x is an element of each of the matrices. For emission cost E and transition costs T, the probabilities P_E and P_T can be computed as $P_E(i,j) = \frac{1}{E(i,j)}$ and $P_T(i,j) = \frac{1}{T(i,j)}$, where $P_E(i,j)$ and $P_T(i,j)$ represents the probabilities of emitting symbol j from state i and transitioning from state i to state j, respectively.

1.3 Correspondence Computation and Pruning

The correspondences between the skeletons are computed by conducting inference on the HMM, resulting in the determination of the most probable sequence of hidden variables, which represents the set of correspondences between two skeletons of their respective point cloud. Chebrolu et al. established correspondence using the computationally intensive Viterbi algorithm. Our approach utilizes an iterative search algorithm rooted in Hidden Markov Models (HMMs) to compute correspondences. Beam search balances exploration and exploitation, efficiently navigating the solution space by prioritizing relevant candidates. This quality renders it preferable over exhaustive search methods like Viterbi,

which may have limitations in certain contexts. Beam search finds extensive use in various fields, including bioinformatics for sequence alignment, object recognition, and image captioning. Refining the search space systematically, guided by the most promising candidates, facilitates progressive convergence. This convergence results in optimal correspondences between point clouds acquired at different time points.
An initial set of correspondences is determined based on beam search and refined based on multiple criteria. The primary selection of a subset of correspondences is based on the Maximal clique estimation. In general, the Maximal clique is **NP-hard** implying the polynomial time exact solution grows exponentially with graph size rendering it impractical for data association. An approximate (maximal clique) solution can however be found by polynomial-time algorithms. The

Algorithm 2 Iterative Beam Search

```
Require: StateNames, Transition Matrix $T$, Emission Matrix $E$, Sequence $V$, Initial
    beam width $b_0$, Beam width reduction factor $r$, Number of iterations $n$, Minimum
    score change $\epsilon$
Ensure: Correspondence pairs
 1: Initialize masks
 2: Initialize $best\_paths\_list$
 3: for $iteration \leftarrow 1$ to $n$ do
 4:   Initialize candidates
 5:   Initialize $best\_paths$
 6:   Set $beam\_width \leftarrow b_0$
 7:   for $t \leftarrow 1$ to size of $V$ do
 8:     Update candidates
 9:   end for
10:   Sort and extract $best\_paths$
11:   Append $best\_paths$ to $best\_paths\_list$
12:   if size of $best\_paths\_list > 1$ then
13:     Calculate $score\_change$
14:     if $score\_change < \epsilon$ then
15:       Terminate
16:     end if
17:   end if
18: end for
19: Choose the best alignment Correspondence pairs
```

algorithm 2, aims to find optimal correspondence pairs between states in a sequence. It takes as input StateNames, Transition Matrix T, Emission Matrix E, Sequence V, Initial beam width b_0, Beam width reduction factor r, Number of iterations n, and Minimum score change ϵ, and outputs Correspondence pairs. The algorithm begins by initializing masks and a list *best_paths_list*. Then, for each iteration from 1 to n, it initializes candidates and *best_paths*, setting the beam width *beam_width* to b_0. For each time step t in the sequence V, candidates are updated based on the current beam width and previous best paths. After updating candidates for all time steps, the algorithm extracts the best paths, appending them to the list *best_paths_list*. If the size of *best_paths_list* exceeds 1, the algorithm calculates the score change and terminates if it falls below the minimum threshold ϵ. Finally, the algorithm chooses the best alignment based on the obtained correspondence pairs and returns them as the output. Throughout the algorithm, parameters such as initial beam width b_0, beam width reduction factor r, number of iterations n, and minimum score change ϵ control the search process and ensure convergence to the optimal solution.

1.4 Temporal Non-Rigid Registration

It is crucial to consider the dynamic nature of plant growth characterized by alterations in both shape and topology while registering temporally separated plant scans. Consequently, to adequately capture these deformations, departing from the conventional assumption of rigidity commonly applied in point cloud registration is essential. The non-rigid changes are characterized by deriving sets of deformation parameters between corresponding skeleton parts of the plant scans. These parameters are estimated using a non-linear least-squares optimization procedure based on the set of correspondences ($C_{S_1 S_2}$), where S_1, S_2 represents a skeleton from two point clouds.
To define the deformations between plant scans, an affine transformation T_i is associated with each node n_i of the skeleton S_1 that undergoes deformation. 3D affine transformation with 12 unknown parameters (3×3 rotation matrix, 3 for translation vector) per node enables the representation of growth and bending by incorporating scaling, shearing, and rotation parameters. The overall objective of non-rigid deformation involves a combination of multiple terms, primarily including correspondence, rotational, and regularization constraints.

1.5 Shape Deformation

The estimated affine transformation parameters deform the entire plant's 3D point cloud by applying a weighted sum of the transformation corresponding to the two nearest nodes. Each point in the deformed cloud is determined based on its proximity to the nearest and second nearest nodes in the skeleton structure, denoted as n_1 and n_2 respectively. Let P_{n1} and P_{n2} represent the transformed positions of P_i using the affine transformations corresponding to nodes n_1 and n_2. The weight factor w (Eqn 1)is calculated as the ratio of the distance between P_i and its orthogonal projection onto the line connecting P_{n1} and P_{n2}, normalized by the distance between P_{n1} and P_{n2}. This weight factor determines the relative position of P_i along the line segment and influences its contribution to the deformation process.

$$w = \frac{\|proj_{p_{n1}p_{n2}}(P_i - P_{n1})\|}{\|P_{n2} - P_{n1}\|} \tag{1}$$

Rotational constraint: The rotational term in the optimization framework quantifies the deviation of the estimated affine transformation from a pure rotation, thereby influencing the smoothness of the deformation.

$$E_{rot} = \sum_{r=1,t=mod(r+1,3)}^{3} (c_r^T c_t)^2 + \sum_{r=1}^{3} (c_r c_r - 1)^2 \tag{2}$$

By incorporating this term into the optimization framework, the extent of shearing is effectively mitigated, ensuring that the resulting deformations maintain a more coherent and natural appearance. Eqn 2 outlines the rotational energy term, where the initial component measures the departure of columns from orthogonality to one another. The subsequent component evaluates the extent of each column's deviation from unit length.

$$E_{reg} = \sum_{t \in N(r)} \|T_r T_t - 1\|_F^2 \tag{3}$$

Regularization constraint: In Equation 3, E_{reg} acts as a regularizing term, ensuring that the transformation parameters of neighboring nodes are similar. This promotes a smooth deformation along the skeleton, resembling the effects of the as-rigid-as-possible constraint. The maximal similarity is aimed for in the affine transformations applied within a surface region. The regularization term is particularly crucial for constraining nodes lacking correspondences. Here, T_r and T_t represent the respective transformations, with t denoting the neighbor of N_r along the skeleton S_2, and F representing the Frobenius norm.