Volume xx (200y), Number z, pp. 1–12

4D Reconstruction of Blooming Flowers

Qian Zheng[1,3†] Xiaochen Fan[2†] Minglun Gong[4] Andrei Sharf[5] Oliver Deussen[2,3] Hui Huang[1,2‡]

[1]Shenzhen University [2]SIAT [3]University of Konstanz [4]Memorial University of Newfoundland [5]Ben Gurion University

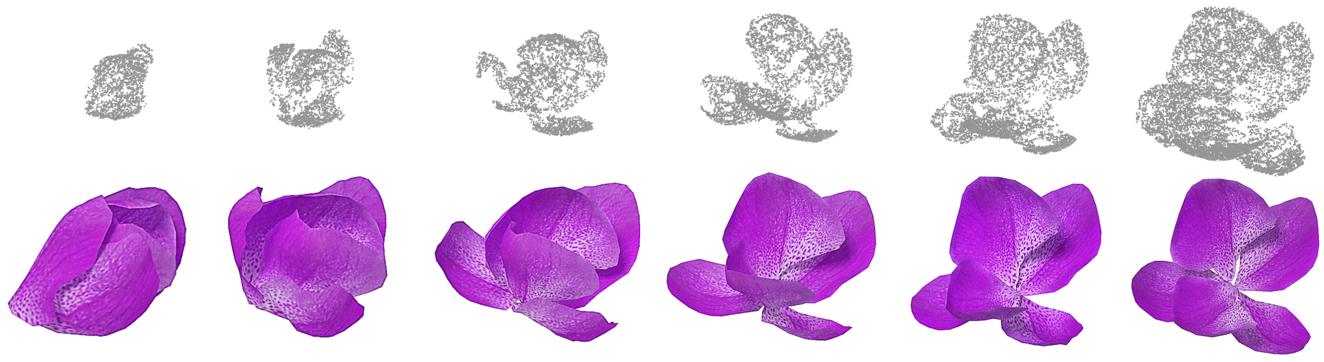

Figure 1: *Reconstruction of a blooming Orchid from a noisy and incomplete point cloud sequence. Note that smaller models representing early stages of the blooming (left ones on the bottom row) are scaled up for a better visualization.*

Abstract

Flower blooming is a beautiful phenomenon in nature as flowers open in an intricate and complex manner whereas petals bend, stretch and twist under various deformations. Flower petals are typically thin structures arranged in tight configurations with heavy self-occlusions. Thus, capturing and reconstructing spatially and temporally coherent sequences of blooming flowers is highly challenging. Early in the process only exterior petals are visible and thus interior parts will be completely missing in the captured data. Utilizing commercially available 3D scanners, we capture the visible parts of blooming flowers into a sequence of 3D point clouds. We reconstruct the flower geometry and deformation over time using a template-based dynamic tracking algorithm. To track and model interior petals hidden in early stages of the blooming process, we employ an adaptively constrained optimization. Flower characteristics are exploited to track petals both forward and backward in time. Our methods allow us to faithfully reconstruct the flower blooming process of different species. In addition, we provide comparisons with state-of-the-art physical simulation-based approaches and evaluate our approach by using photos of captured real flowers.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Capturing the formation and growth of plants is interesting for animation applications in advertising, film, and games, but also very important for studies in botany, forestry, and agricultural science. It is of great significance to both flower simulation and botanical research, enabling novel observations, viewing angles, and analysis into the process. Plants are most volatile during their early stages. As new strains appear, their behavior may consistently vary and thus it is important to analyze their growth at these stages. Traditional studies mainly rely on manual recordings of growth stages, or image-based measurements taken at sparse intervals. Such workflows are tedious, prone to measurement bias, and difficult to scale to large-scale observations, both in space and time.

Advances in affordable 3D acquisition devices now provide new opportunities in capturing and modeling 3D real-life phenomena in time. For instance, Li et al. [LFM*13] proposed a method for detecting bifurcations of indoor plants from 4D point cloud data. The problem of reconstructing 4D sequences of blooming flowers, however, is considerably more challenging. During the flower opening, inner petals, which were previously occluded, appear and commence an intricate deformation process. Throughout this process, petals may stretch, bend, and finally shrink and wrinkle, while

† Joint first authors

‡ Corresponding author: Hui Huang (hhzhiyan@gmail.com)

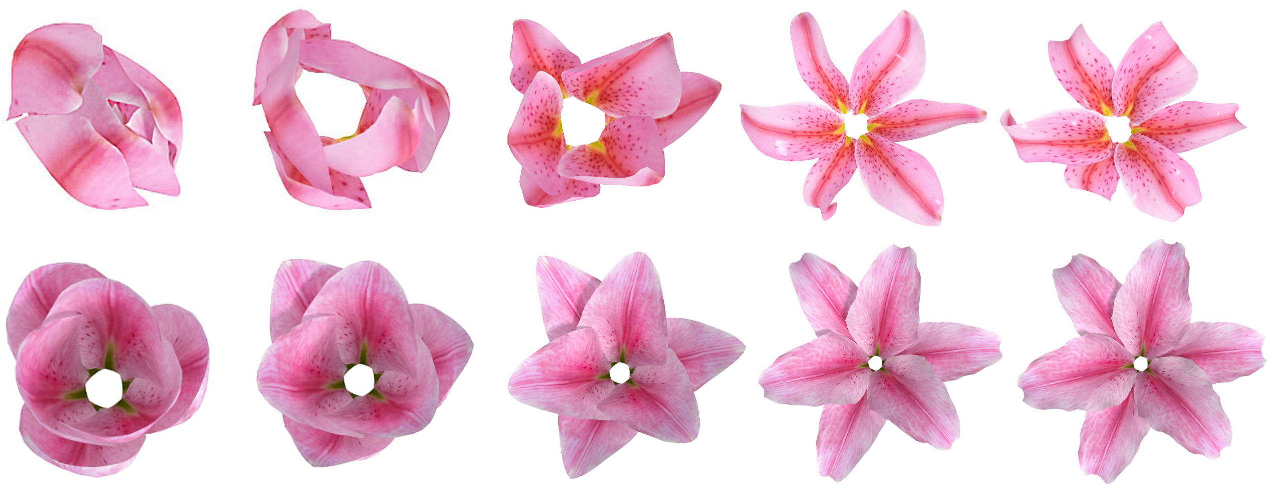
Figure 2: *Side-by-side comparison between our modeling results (top) and the results from the state-of-the-art flower blooming simulation [LLX*15] (bottom). While the simulated results look more regular, our data-driven approach captures botanic reality.*

possibly colliding with other flower structures. Although approaches have been proposed to create 3D animations of flower openings by simulating petal deformations based on physical and botanical properties [IYKI08, LLX*15], the synthetic animation sequences generated do not offer realistic petal shapes nor natural petal motions; see Figure 2 (bottom).

We tackle this problem by proposing a new template-based dynamic tracking algorithm, which takes into consideration the specific characteristics of flowers. Having a full flower geometry as a template, we track and fit it to the point cloud sequence on a frame by frame basis. Template-based tracking methods so far focus on faces, hands, bodies, and other deformable objects, but none of these works aims at tracking flowers with multiple petals showing heavy occlusions and collision-based interactions. To address these challenges, we introduce additional priors that assist the tracking process, including petal deformation energy, collision avoidance, as well as boundary and root position constraints.

In summary, our main contributions are as follows:

- a technique for reconstructing flower petals that gradually appear in a sequence of point clouds. This correctly infers the shape of occluded petals, which are in general missing from the scanned data at early blooming stages.
- a flower deformation optimization which enforces minimum deformation and collision avoidance constraints. This accurately reconstructs individual petals.
- applications that demonstrate the usability of reconstructed mesh sequences of blooming flowers.

2. Related Work

We summarize previous works in the realm of plant modeling and dynamic reconstruction.

Plant modeling. We refer to [DL10] for a comprehensive survey on previous plant modeling techniques. While it is possible to create very realistic looking static flowers using L-systems [PL96] or other modeling methods, we are interested in capturing both form and dynamics of real flower blooming. This is very cumbersome using L-Systems [PHM93]. It can be done more easily with parametric approaches [LD99], which are still not data-driven, though.

Ijiri et al. [IOOI05] introduce a sketching interface for flower editing, which preserves botanical structures but does not cover their animation. To model 3D flowers from real-world samples, Yan et al. [YGCO*14] reconstruct flower petals from a single photo by using surfaces of revolution. This approach uses a number of assumptions and also does not animate the models. Zhang et al. [ZYFY14] address the challenging task of modeling static 3D flowers from imperfect point clouds. They build morphable models of petal shapes from multiple exemplars in a library and fit them to a given point cloud. Our method works without a pre-given library of petals and instead uses single thin-plate meshes as templates obtained through simple user interaction. Ijiri et al. [IYYI14] propose a semi-automatic modeling technique to reconstruct complex 3D flowers that were entirely captured by X-ray tomography (CT). They use an active contour model composed of shaft and sheet primitives to reconstruct flower stems and petals by optimizing an energy functional. This setup might probably be too expensive for most applications and thus want to develop a reproducible solution.

Most recently, Xie et al. [XYS*16] introduce an interactive tree modeling system that utilizes examples of real tree-cuts to enhance tree creation with realistic structures and fine details. Yin et al. [YHL*16] propose a template-based method to reconstruct full 3D plant models from incomplete point clouds [HWCO*13]. They build individual leaf templates by cutting leaves away from a plant and then register together these disjoint parts. While this approach could be applied for generating a static flower model from its individual petals, it requires cutting off the petals and hence cannot capture the whole flower blooming process.

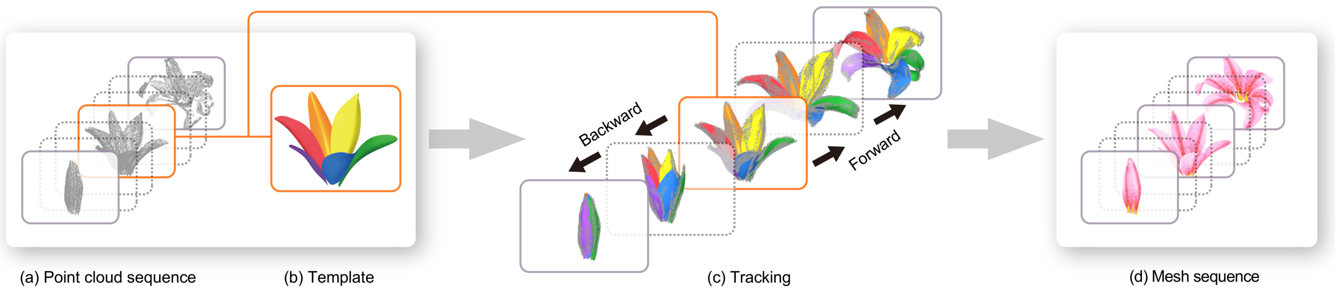

Figure 3: *Algorithm overview. We scan a blooming flower to generate a sequence of point clouds (a). The full flower geometry is then reconstructed at a manually selected intermediate time step (b). We track and fit the reconstructed flower petals both forward and backward in time using a template-based dynamic tracking algorithm (c). A spatially and temporally coherent 3D model sequence of a flower blooming is thus successfully reconstructed (d).*

Modeling of plant growth. The morphology of plants and flowers changes significantly during growth. Due to the inherent geometrical complexity of this process, only few works in computer graphics track and simulate such morphological changes.

Botanists have tracked growth of petal shapes using clonal analysis [RLBC03, CCG*10, GKH*10]. They observed that flowers and leaves have similar mechanics that govern their growth as well as their decay [LM09, LM11, XC11]. Change is driven by non-uniform stretching and shrinkage forces, which can be simulated by accounting for measured differential properties. The curled shape of drying leaves is caused by non-uniform and local changes in volume [CRLM*04]. Xiao and Chen [XC11] use measured strains of dried leaves to derive a linear differential strain field and model leaf bending by shrinkage. These simulation models are very complex and thus lack of efficiency and validation on accuracy.

Several works attempt to recover the geometry of growing plants by integrating multiple measurements of a real-life exemplars [MEL*05, FDM*10]. We refer the reader to a comprehensive survey [PR12] for more details. In contrast to these works, we scan the flower geometry and reconstruct it directly.

In computer graphics, the simulation of blooming flowers has been introduced by Ijiri et al. [IYKI08]. They use an elastic triangular mesh to represent petals and simulate their growth in a semi-automatic way according to user-specified parameters.

To simulate realistic shape deformation of drying leaves, Jeong et al. [JPK13] assume that shrinkage depends on water changes and thus simulate the internal water loss for controlling the shrinkage of leaves. Li et al. [LLX*15] simulate flower blooming by representing flower petals with an elastic triangular mesh and a growth curve, which allows the user to control the global bending of petals. We avoid such complex biological models by directly tracking and reconstructing real-life flowers from scanned point clouds.

Similar to us, Li et al. [LFM*13] reconstruct the growth and development of plants acquired by 3D scanning devices [HLZ*09, HWG*13]. They track topological events like plant budding and bifurcations through a forward-backward analysis of time-lapse point clouds. As opposed to simple plants, blooming flowers undergo a much more complex process with respect to their shape deformations, petal collisions and inter-occlusions.

Our work is also related to the technique [LDS*11], which generates moving trees from video. Pirk et al. [PSK*12] leverage dynamic tree modeling and representation for interactively adapting complex tree models to their environment. Pirk et al. [PNDN12] compute developmental stages of a tree from a static input model. The focus of these works is, however, not on geometric details.

Non-rigid tracking. Tracking surfaces in dynamic point clouds is a well-studied topic. The majority of papers focuses on human performance capture for faces [KRP*15], hands [KA14, QSW*14, TST*15], bodies [ZSZ*14] and other articulated objects [SNF14]. Nevertheless, none of these works aims at tracking plants and flowers due to the specific complexity of these sequences.

Tracking deformable objects has been previously addressed in several works: Bojsen-Hansen et al. [BHLW12] present a method for recovering a temporally coherent, deforming triangle mesh with arbitrarily changing topology from an incoherent sequence of meshes. Schulman et al. [SLHA13] track a deformable object in a point cloud based on a probabilistic generative model that uses a physical model of the tracked object. Similarly, the work of Wang et al. [WWY*15] tracks a soft object, such as a Lotus leaf, from sparse point clouds with the guidance of a physical model.

Our method is inspired by these combinations of templates and physical models. In contrast to tracking single objects, we aim at tracking *multiple* petals with heavy occlusions and collisions. Due to subtle physical processes that occur during the blooming process it is infeasible to capture the complexity of a blooming flower using a simple physical model. Therefore, we take a geometric approach aiming at accurate reconstructions from scanned data.

3. Overview

We capture a blooming flower by scanning it over time and generating a sequence of 3D point clouds. During the blooming process, the points sample the visible shapes of flower petals at different time steps, which are denoted by $\mathbb{Q} = \mathbb{Q}_{1:T}, 1 \leq t \leq T$, where t

represents time. At early blooming stages, interior petals are completely occluded; therefore they cannot be sampled.At later stages, exterior petals decay, bend, and twist heavily; therefore they might also only be poorly sampled (Figure 3(a)).

Because shapes of individual petals are best captured at an intermediate stage of the blooming process, we manually select an intermediate key frame from the sequence to reconstruct the full geometry of the flower (Figure 3(b)). We denote the whole flower by M, and the geometry for each petal k ($1 \leq k \leq K$) by $\mathbb{M}^k$. To build the full geometry, we use an interactive skeleton-driven modeling technique [YHZ*14]. Specialized petal modeling methods from incomplete point clouds, as demonstrated in [ZYFY14], could also be applied here. Please note that we assume all petals being at least partially visible at the selected key frame, which is a reasonable assumption for most flowers.

We use a full 3D flower model as a template and generate its petals during the time by deforming and fitting it to the point clouds in adjacent time steps. Our *track-and-fit* algorithm is formulated as a Maximum a Posteriori (MAP) problem and solved using Expectation Maximization (EM) [DLR77]. During the expectation step, correspondences between the template mesh and the captured points are established, which are then used for guiding mesh deformation in the maximization step. We track flower petals both forward and backward in time (Figure 3(c)), yielding a spatially and temporally coherent 3D model sequence $\mathbb{M}_{1:T}, 1 \leq t \leq T$, which represents the flower blooming process well (Figure 3(d)).

To address the specific challenges of tracking flowers, we use the popular MAP formation [MS10, SLHA13, WWY*15] for tracking and registration, but utilize petal relations for both establishing correspondences and deforming the mesh. To establish correspondences, we classify the points into different parts and consider boundary constraints between points and vertices. To deform the template, we enforce a petal shapes preservation term and a petal penetration avoidance term. The first term regularizes the shape of each petal, while the second regularizes the configuration between petals. Because of this, our tracking result at each time step is a realistic flower model even if the point cloud is incomplete.

4. Method

During tracking, the topology of our template meshes remains unchanged, but the locations of their vertices need to be updated to match the captured point clouds. In this section we first describe the acquisition of our data, then we discuss how to compute a mesh model $\mathbb{M}_t$ based on a point cloud $\mathbb{Q}_t$ captured at time t and an already computed mesh model $\hat{\mathbb{M}}$, where $\hat{\mathbb{M}} = \mathbb{M}_{t-1}$ for forward tracking and $\hat{\mathbb{M}} = \mathbb{M}_{t+1}$ for backward tracking.

4.1. Data acquisition

The duration different flower species need for blooming varies from hours to days. Hence, scanning the whole process requires a robust acquisition setup. Similar to [LFM*13] we put the flower on a turntable and use a structured light scanner to capture the flower shape from different sides using a fixed scanner location (see Figure 4(a)). We assume the blooming process to be slow enough for

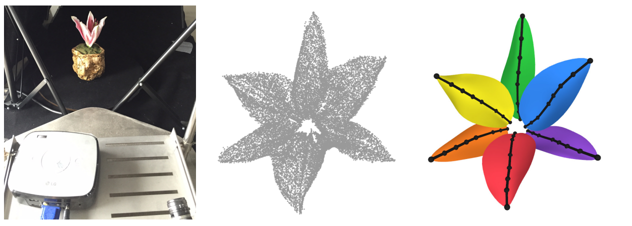
(a) System setup (b) Point cloud (c) Template

Figure 4: *Using a structure light scanner (a), we obtain a point cloud sequence capturing a blooming flower. From the sequence, we select a point cloud (b), which provides a rather complete data sampling, and reconstruct a mesh template (c) for it. The black lines are curves skeletons of the petals.*

registering point clouds captured under different turntable settings under a rigid transformation. Nevertheless, rotating flowers using a turntable causes small perturbations of their shapes, which results in noise and outliers within the registered 3D point clouds. We choose to handle this at the later stage rather than relying on more advanced and expensive acquisition setups.

4.2. Defining a Maximum a Posteriori Problem

We consider a point cloud $\mathbb{Q}_t$ as the observation of the Gaussian Mixture Model, whose centroids are the vertices of the unknown mesh $\mathbb{M}_t$. We solve for these GMM centroid locations by formalizing it as a Maximum a Posteriori (MAP) problem:

$$\operatorname*{argmax}_{\mathbb{M}_t} p(\mathbb{Q}_t|\mathbb{M}_t)p(\mathbb{M}_t), \qquad (1)$$

where the first term $p(\mathbb{Q}_t|\mathbb{M}_t)$ is the likelihood term, representing the probability of having the points of the point cloud under the assumption of the model, and the second term $p(\mathbb{M}_t)$ is the prior term, representing the probability of the model to exist.

Taking the model obtained at the previous frame $\hat{\mathbb{M}}$ as a starting point, the EM algorithm is employed to solve the MAP estimation problem. The Expectation-step (E-step) estimates the latent variables $\mathbf{Z}$, which represent correspondences between the point cloud $\mathbb{Q}$ and the vertices of the current model $\mathbb{M}$, where $\mathbb{M}$ is set to $\hat{\mathbb{M}}$ initially. The Maximization-step (M-step) updates the vertex positions in $\mathbb{M}$ by solving an energy minimization problem based on the current correspondences $\mathbf{Z}$. We now describe the two steps in details and elaborate how to reach to an optimal solution.

4.3. E-step: Estimation of Correspondences

Given a mesh model $\mathbb{M}$, our task within the E-step is to find the correspondences between vertices in $\mathbb{M}$ and and the point cloud $\mathbb{Q}$ captured at the current time. To improve accuracy, we first classify the points in $\mathbb{Q}$ into different parts, each part $\mathbb{Q}^k$ corresponding to a petal mesh $\mathbb{M}^k$. We then estimate the correspondences between points in $\mathbb{Q}^k$ and vertices in $\mathbb{M}^k$ for each petal k.

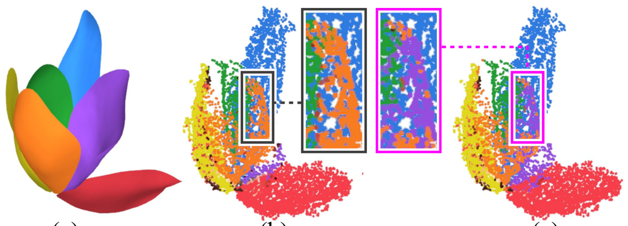

Figure 5: *Using only the Gaussian Mixture Model of the previous mesh (a), a fast moving (purple) petal, yields an incorrect point classification (b) as some points are mistakenly assigned to the wrong (orange) petal. Utilizing additional priors for the petal structure helps to solve this problem (c).*

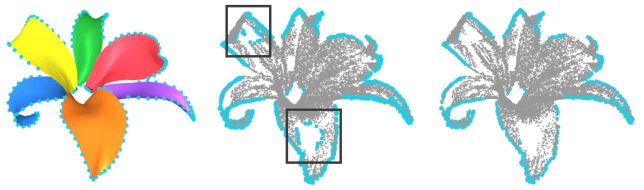

Figure 6: *Cyan points in (a) show the boundary of the current mesh template. In (b) candidate boundary points in the scan are detected using local features, which may contain outliers (highlighted with black boxes). In (c) boundary points outliers are removed in the scan using the current mesh model.*

Point cloud classification. Under the GMM assumption, a captured point in $\mathbb{Q}$ is most likely associated with the closest petal. Each vertex m_i in $\mathbb{M}$ represents a Gaussian centroid that is associated with a diagonal covariance matrix Σ_i, as well as a weight ϕ_i measuring the impact of m_i with respect to the observation. We set Σ_i to be the average distance of m_i to its one-ring neighborhood along x-axis, y-axis, and z-axis, respectively. The weight ϕ_i is set to be 1 initially and updated after estimating correspondences.

A point q_j of the point cloud is normally distributed around m_i as $q_j \sim \mathcal{N}(m_i, \Sigma_i)$. The probability of q_j given m_i is given by:

$$p(q_j|m_i) = \frac{1}{\sqrt{(2\pi)^3|\Sigma_i|}}\exp(-\frac{1}{2}(q_j-m_i)^T\Sigma_i^{-1}(q_j-m_i)).$$

For a point q_j, its probability of belonging to $\mathbb{Q}^k$ is computed as:

$$p(q_j \in \mathbb{Q}^k) = \frac{\sum_{m_i \in \mathbb{M}^k} \phi_i p(q_j|m_i)v_i}{\sum_{m_i \in \mathbb{M}} \phi_i p(q_j|m_i)v_i}, \tag{2}$$

where numerator and denominator represent the probabilities computed from the GMMs. The visibility term $v_i \in \{0,1\}$ has a value of 1 if v_i is visible, i.e., can be seen from the viewpoints of the scanner. We only assign point q_j to $\mathbb{Q}^k$ if there is a large probability for this, i.e., $p(q_j \in \mathbb{Q}^k) \geq 0.99$ in practice. As a result, noisy points that are far away from the mesh surface will not be classified into any parts.

The initial classification is further refined based on a priori knowledge on petal structures. First, when two petal meshes overlap each other, an observed point likely belongs to the exterior petal, i.e., the petal is fully visible with one side at the current time step. Hence, if petal k is detected as the exterior petal based on mesh $\mathbb{M}^k$, a point q_j is assigned to $\mathbb{Q}^k$ if its distance to any visible vertex m_i of $\mathbb{M}^k$ is within Σ_i. Furthermore, if the location of a petal $\mathbb{M}^k$ converges after several EM iterations, for any not-yet associated point outside the Σ_i distance of $\mathbb{M}^k$, its probability of belonging to a petal is computed with $\mathbb{M}^k$ being ignored.

Point correspondence. With points in $\mathbb{Q}$ labeled by their corresponding petals, we estimate the correspondences between points in $\mathbb{Q}^k$ and vertices in $\mathbb{M}^k$ for each petal k. Correspondences are represented using an association matrix $\mathbf{Z} : (\mathbb{M}^k \rightarrow \mathbb{Q}^k)$. Each entry $\mathbf{Z}_{ij} \in [0,1]$ indicates how likely point q_j corresponds to vertex m_i:

$$\mathbf{Z}_{ij} = \frac{\phi_i p(q_j|m_i)v_i}{\sum_{m_i \in \mathbb{M}^k} \phi_i p(q_j|m_i)v_i}, \qquad \text{with} \quad \phi_i = \sum_{q_j \in \mathbb{Q}^k} p(q_j|m_i), \tag{3}$$

where the numerator and denominator are the corresponding probabilities computed from the Gaussian centroid and the petal GMM.

Boundary correspondence. Our deformable mesh models so far cannot be enforced to follow the contour of the flower petals, since only point correspondences are used (cf. Figure 7(a)). Hence, we have to use additional constraints by looking for correspondences between point cloud boundaries and mesh boundaries. We denote the boundary vertices by $\mathbb{Q}_\mathbb{B}$ with $\mathbb{Q}_\mathbb{B} \subset \mathbb{Q}$ and boundary points by $\mathbb{M}_\mathbb{B}$ with $\mathbb{M}_\mathbb{B} \subset \mathbb{M}$ (Figure 6(a-c)).

Since each petal k is modeled using a single layer triangle mesh $\mathbb{M}^k$, its vertices along the edges can be easily detected. To avoid matching ambiguities, we here only add a vertex m_i to the boundary vertex set ${\mathbb{M}_\mathbb{B}}^k$ if its distance to other petals is larger than a given threshold (set to $2 \times \Sigma_i$).

We first detect the boundaries of the whole point cloud $\mathbb{Q}$ using local features [GWM01]. For a point q_j, its local neighbors within a radius r are then determined and projected to a locally fitted plane. Next, an 1D polar histogram of 12 bins is computed, which measures the directions of projected local neighbors on the fitted plane using q_j as the origin. If more than 20% of the bins have zero value, q_j is considered as a candidate for a boundary point. These candidates for boundary points are then filtered using the current mesh model $\mathbb{M}$, which removes outliers that typically appear along the boundary of interior holes (Figure 6(b)). A point q_j is a boundary point ($q_j \in \mathbb{Q}_\mathbb{B}$) if any of its k-nearest vertices in the mesh is a boundary vertex. Finally, the detected boundary points are labeled by the petal that they belong to (denoted by ${\mathbb{Q}_\mathbb{B}}^k$). Once $\mathbb{Q}_\mathbb{B}$ and $\mathbb{M}_\mathbb{B}$ are fully determined, the association matrix $\mathbf{Z} : ({\mathbb{M}_\mathbb{B}}^k \rightarrow {\mathbb{Q}_\mathbb{B}}^k)$ between the two sets is calculated using Equation (3).

4.4. M-step: Updating vertex locations

Having the correspondences between the mesh vertices and captured 3D points, the M-step optimizes the vertex locations in $\mathbb{M}$

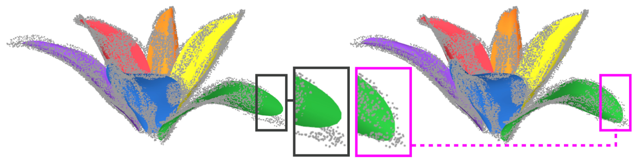

Figure 7: *Deforming the mesh template using point correspondences alone cannot effectively enforce the mesh to fit the point cloud (left). Adding an additional penalty on distances between mesh and point boundaries solves the problem (right).*

and thus allows $\mathbb{M}$ to better fit the data $\mathbb{Q}$. This is achieved by using

$$\operatorname*{argmin}_{\mathbb{M}} \left(-\log p(\mathbb{M}|\mathbb{Q},\mathbf{Z}) - \log p(\mathbb{M})\right), \tag{4}$$

where the first term, the *data term* reflects how well the model explains the point cloud, and the second term, the *prior term* regularizes the solution to ensure a realistic flower model.

4.4.1. Data term

The data term is defined based on the distances between mesh vertices and their corresponding points. Hence, when the data term is minimal, the mesh is well aligned with the captured points. To ensure that the mesh also follows the contour of the petal, an additional penalty is given to distances between corresponding vertices and points along the boundaries:

$$-\log p(\mathbb{M}|\mathbb{Q},\mathbf{Z}) = \sum_k \left(w_1 D(\mathbb{Q}^k, \mathbb{M}^k) + w_2 D({\mathbb{Q}_{\mathbb{B}}}^k, {\mathbb{M}_{\mathbb{B}}}^k)\right), \tag{5}$$

where the distance function is defined as:

$$D(\mathbb{M},\mathbb{Q}) = \sum_{m_i \in \mathbb{M}} \sum_{q_j \in \mathbb{Q}} \mathbf{Z}_{ij}(q_j - m_i)^T \Sigma_i^{-1}(q_j - m_i), \tag{6}$$

with weights w_1 and w_2 set by default to be 0.02 and 0.05.

Figure 7 demonstrates the importance of the boundary term to ensure that the reconstructed mesh closely follows the scanned boundary. Typically, we are not able to detect any boundary points at the very beginning of the blooming process since different petals are not sufficiently separated. In this situation we have $\mathbb{Q}_{\mathbb{B}} = \emptyset$ and hence the boundary term does not affect the above optimization.

4.4.2. Prior terms

To enhance the robustness of the tracking process against noise in the data and to ensure that the tracked mesh looks like a flower, additional constraints are applied based on the properties of the flower petals. We define three terms for this: petal shape preservation, penetration avoidance, and fixed petal root position. The overall prior term is thus given by:

$$-\log p(\mathbb{M}) = E_{\text{shape}} + E_{\text{collision}} + E_{\text{root}}. \tag{7}$$

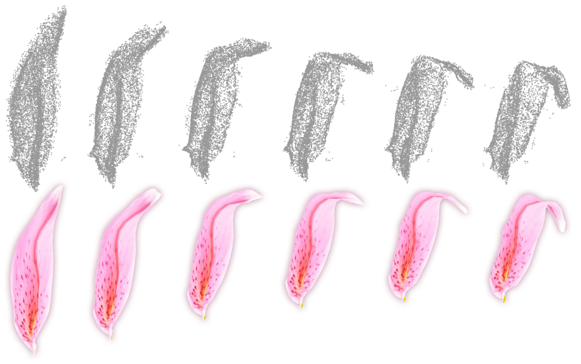

Figure 8: *Tracking and fitting a single petal. The captured point clouds at different time steps (top) are tracked and fit by a petal template model (bottom) using data fitting and shape preservation.*

Shape preservation. To ensure that the generated meshes look like flower petals, a previous work on static flower petal modeling [ZYFY14] uses morphable models generated from a petal database. In contrast to them, we let the user model a template mesh and assume this model can faithfully represent the flower. Hence, we would like to constraint the petal shapes for all time steps to have similar shapes as the created template. This is achieved by deforming the template using solely control points that are located on the petal skeleton and the boundary curves; see Figure 4(c). It is worth noting that when generating flower animations manually, artists usually also manipulate these three curves.

The affine transformations of all control points, denoted as $\mathbb{T}$, determine the location of the mesh $\mathbb{M}$ by using a linear blend skinning (LBS) model. This model is expressed in matrix form: $\mathbb{M} = G\mathbb{T}$, where G stands for a large matrix that combines the original positions of the template with blending weights associated with control points. The blending weights are calculated using harmonic coordinates [JBPS11]. For the tracking, we actually recover $\mathbb{T}$ to match the data terms and prior terms.

As flower growth between two adjacent time steps is almost unnoticeable, we aim at restricting the deformation to isometry, preferring the shape being bent rather than stretched. We do not require isometry for the whole sequence though, as the petals can grow bigger in time. The shape distortion is measured by the as-rigid-as-possible energy term [SA07, JBK*12]:

$$E_{\text{shape}}(\mathbb{M},R) = w_3 \sum_i \sum_{j \in \mathcal{N}(i)} c_{ij} \|(m_i - m_j) - R_i(\hat{m}_i - \hat{m}_j)\|_2^2, \tag{8}$$

where $\mathcal{N}(i)$ is the one-ring neighborhood of m_i, $\hat{m}_i$ represents the vertex position obtained at the previous frame, c_{ij} is the familiar per-edge cotangent weight, and $R = \{R_i\}$ with R_i being the unknown local rotation of the m_i. Interested readers are referred to [SA07] for more details. The weight w_3 is set to be 1 by default.

Figure 8 demonstrates that these energy terms combined with the control points are able to accurately recover the motion of a single petal during the blooming process.

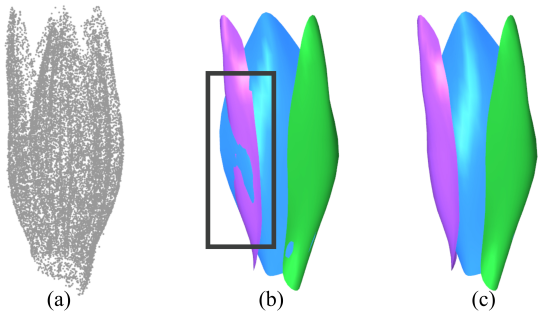

Figure 9: *At early stages of blooming (a), internal petals are completely occluded by external ones and tightly packed. Since there are only a few sampling points of internal petals, tracking and fitting the data here may result in large petal intersections (b). We apply the penetration avoidance term to effectively prevent it (c).*

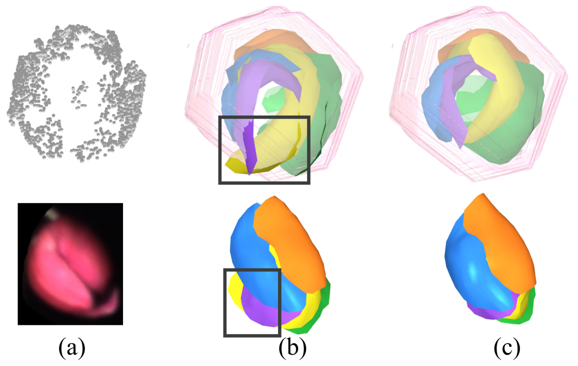

Figure 10: *At early blooming stages, the scanned points capture only the exterior petals and tips (top-left). Reconstructed petals may thus penetrate the visual hull (mid column). Constraining petals from penetrating the visual hull (right column) leads to results closer to the observed photo (bottom-left).*

Penetration avoidance. The first prior forces the individual petals to have proper shapes. Next, we additionally have to prevent them from colliding with each other. This is done by setting up a penetration avoidance prior. Again, we assume that the template mesh provides correct spatial relations among different petals. If we detect that a vertex m_i penetrates a petal, we shall push m_i so that the penetration is avoided. This is realized by first finding m_i's closest point p with normal n in the petal and then move m_i to $\overline{m}_i$ so that $n^T(\overline{m}_i - p) < 0$ is satisfied. In practice, we set $\overline{m}_i = p - 1.05n\|m_i - p\|_2$.

Furthermore, when tracking backward to earlier stages, the interior petals are almost completely hidden and so they cannot be constrained by a data term. We thus employ an observed *visual hull* [Lau94], the intersection of visual cones derived from different turntable settings, to further constrain the range of movements of petals. If we detect a vertex m_i that penetrates the visual hull, we compute its desired position $\overline{m}_i$ as shown above.

Based on the detected penetrating vertices and their desired correct positions, the penetration prior term is now computed as:

$$E_{\text{collision}}(\mathbb{M}) = w_4 \sum_{m_i \in S_C} \|m_i - \overline{m}_i\|^2, \tag{9}$$

where S_C is the set of vertices penetrating other petals or the visual hull, and weight w_4 is set by default to be 2.

Figures 9 and 10 demonstrate that the penetration avoidance term can effectively resolve collisions between different petals, as well as avoiding the petals go beyond the visual hull.

Fixed root position. During blooming, the root point of each petal usually remains stationary relative to the flower head. Hence, by specifying a root vertex for each petal in the template mesh we define a root prior term as:

$$E_{\text{root}} = w_5 \sum_{m_i \in S_R} \|m_i - \hat{m}_i\|^2, \tag{10}$$

where S_R is the set that contains all root vertices. In practice, we want this term to provide a strong constraint and therefore use a high weighting value with $w_5 = 100$.

4.5. Energy Minimization

Since flower blooming is a gradually changing process, the tracking procedure only needs to search local optima in the neighborhood of the existing mesh model. We obtain the local optimal solution for Equation (1) using the EM algorithm. The E-step estimates the correspondences $\mathbf{Z}$ based on the vertices of the current model $\mathbb{M}$, and M-step updates the vertex positions by solving an energy minimization problem (Equation (4)). EM iterations stop when the ratio of energy changes of subsequent steps is smaller than a given threshold, for instance, 10^{-3}.

Equation (4) is a non-linear least squares problem as E_{shape} (Equation (8)) and $E_{\text{collision}}$ (Equation (9)) are non-linear. We compute its local optima using the *projection-based* iteration method proposed in [BDS*12, BML*14]. In the local step, the rigid transformation R_i for any vertex in E_{shape}, as well as $\overline{m}_i$ in $E_{\text{collision}}$, is updated separately using current vertex positions. In the global step, we solve a linear system to find the least square solution for the transformations $\mathbb{T}$, where the vertex positions are updated accordingly.

5. Results and applications

A variety of blooming flowers have been captured and tested in our experiments. To demonstrate the generality and robustness of our algorithm, we focus on flowers of different sizes, complexities, blooming rates and shapes.

To reconstruct the flower blooming sequence, we track and fit the reconstructed petal templates using the above-mentioned EM optimization. This involves computing local deformations and estimations for the correspondences for each petal independently, which simplifies the computation and allows our algorithm to run at fast speed. For an average dataset with 20K scanned points per frame and 200 vertices per reconstructed petal mesh, our algorithm takes less than 1 second for each EM iteration on an Intel Core i7 CPU @3.40GHz with 24GB RAM. The processing time is about 10 seconds per frame and about 20-30 minutes for the whole sequence;

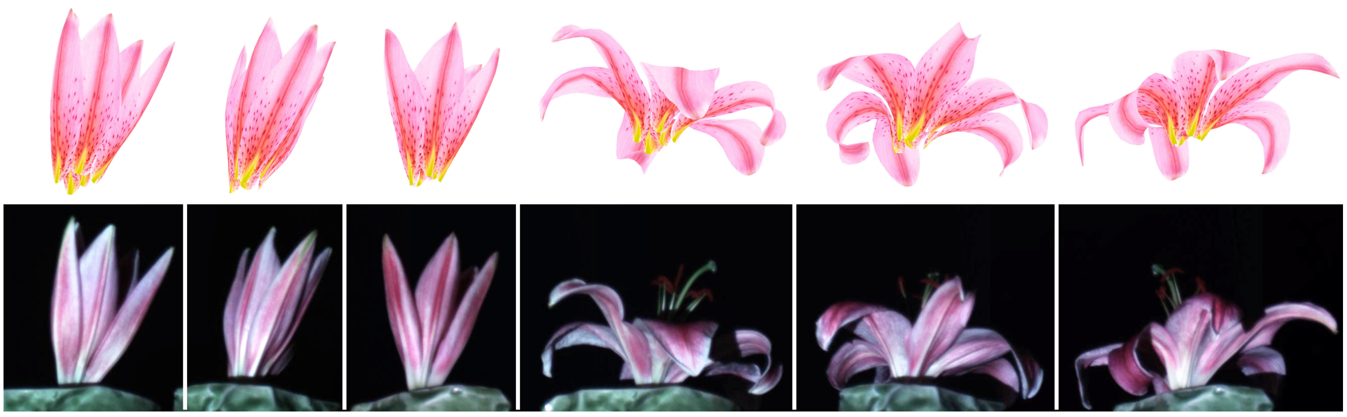

Figure 11: *Comparison between our reconstructed models (top) and captured image sequence (bottom) of a blooming Lily flower. Note that different viewpoints are used at different blooming stages and the filaments are not modeled in our approach.*

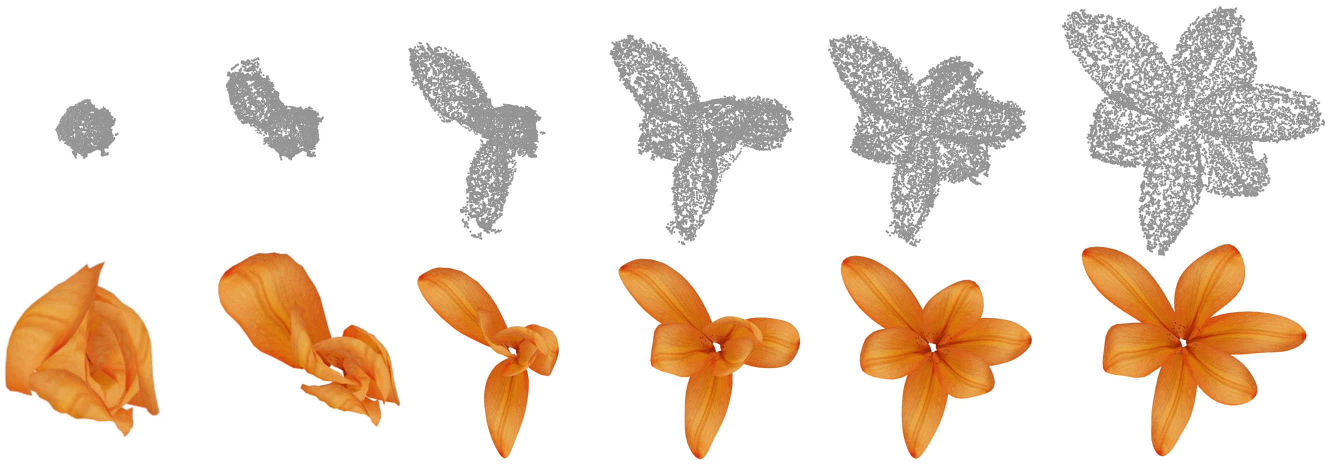

Figure 12: *The blooming sequence of a Golden Lily. At early stages, the petals have complex shapes and packed in a small space. Hence, they are poorly represented in the captured point clouds (top). Nevertheless, our approach recovers the full model sequence (bottom).*

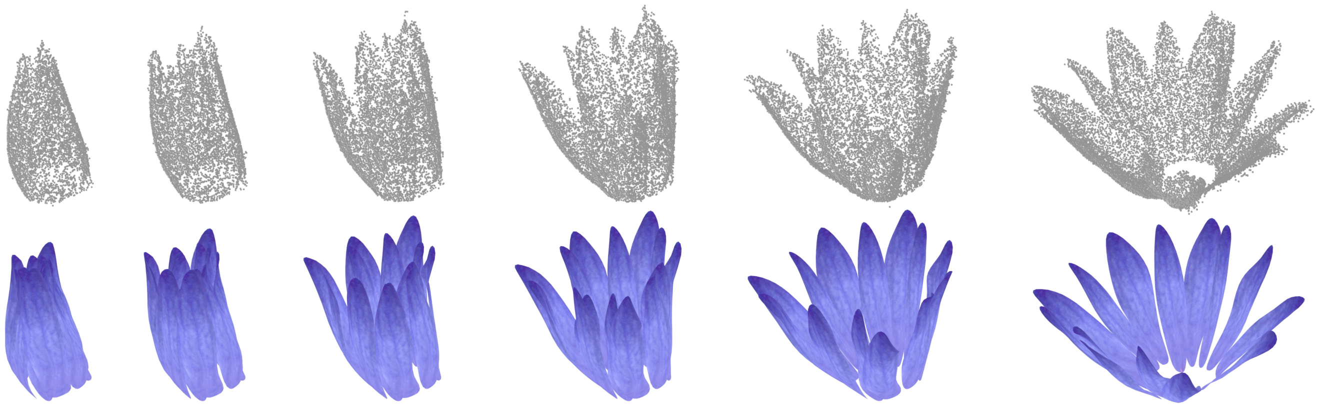

Figure 13: *An opening Water Lily consists of narrow and thin petals. As a result, it is hard to separate different petals from the scan data (top). Our approach is able to reconstruct the complete sequence (bottom) through tracking a template model.*

Sequence	#Days	#Petals	#Frame	Time
Orchid (Fig. 1)	5	5	114	6.0
Lily (Fig. 11)	5	6	128	9.4
Golden Lily (Fig. 12)	5	6	111	10.7
Water Lily (Fig. 13)	3	14	73	9.8
Eustoma (Fig. 15)	5	5/8	123	5.6

Table 1: *Average processing time (rightmost column) in seconds per frame, along with data information (e.g., blooming rate in leftmost column) for our testing sequences.*

Sequence	w_1	w_2	w_3	w_4	w_5
Orchid (Fig. 1)	0.002	0.005	1	2	100
Lily (Fig. 11)	0.01	0.02	1	2	100
Golden Lily (Fig. 12)	0.005	0.005	1	2	100
Water Lily (Fig. 13)	0.01	0.005	1	2	100
Eustoma (Fig. 15)	0.01	0	1	2	100

Table 2: *Parameter settings for all examples presented in this work.*

see Table 1. Please note that compared with the data capturing time, governed by how long the flowers need for blooming, our processing time is negligible.

Our method uses a set of 5 parameters $\{w_1, w_2, w_3, w_4, w_5\}$ controlling the 5 terms of our constrained optimization. Table 2 summarizes the parameter settings in our experiments. Overall, the parameter tuning ranges are small and the parameters do not vary much during our experiments. In fact, the same weights are used in all examples for shape preservation (w_3), penetration avoidance (w_4), and root position (w_5) constraints. The ratio between w_1, w_2 (point and boundary alignment) and w_3 is mainly determined by the ratio between the number of points and the number of vertices. In particular, the specific values of w_1 and w_2 are mostly affected by the quality of the input data. If 3D point positions incorporate large amounts of noise (e.g., those of Orchid and Golden Lily), we apply a smaller w_1, otherwise the tracked flower might be overly fitted and distorted. Similarly, we reduce w_2 when the captured boundaries of petals are not reliable.

Next, we discuss different aspects of our results and provide snapshots zooming in and focusing on different characteristics.

In Figure 1, we show the 3D reconstruction sequence of a blooming Orchid. Due to self-occlusions, the acquired point cloud is noisy with large missing parts at occluded regions. Note that our method is able to recover the complete Orchid model including the fine geometry and location of petals.

Figure 2 compares our method with a state-of-the-art physics-based blooming simulation method [LLX*15]. The comparison highlights the simple and symmetric looking of pure physics-based modeling results, whereas our data-driven approach produces more realistic and versatile animations.

Figure 3 illustrates a blooming sequence of a Lily. Highly cluttered petals at early stages of the sequence are poorly sampled. Using our forward and backward tracking and the fitting algorithm, we are able to reconstruct even the early stages with proper 3D models. At late stages, the petals undergo heavy deformations of twisting and wrinkling. Our template model deforms and fits accurately these highly curved petals.

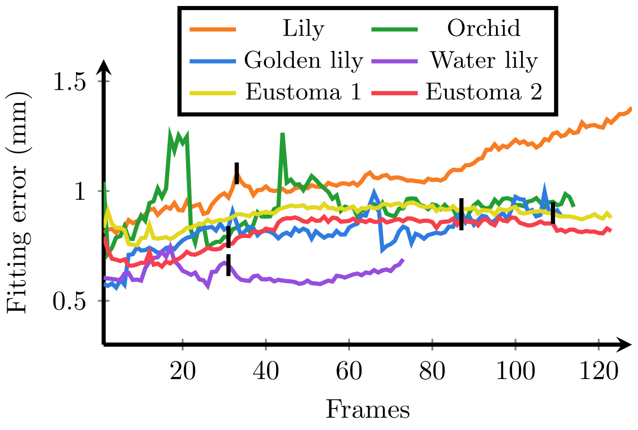

Figure 14: *Reconstruction errors of all examples. Vertical black bars indicate the fitting errors of the pre-built templates.*

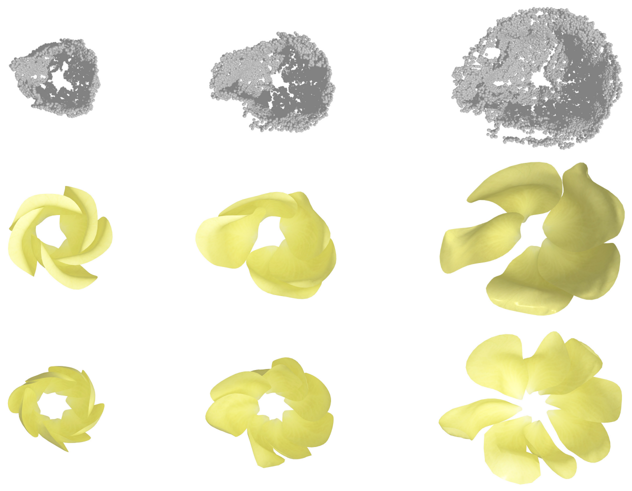

Figure 15: *Using different petal templates, we can generate two flower blooming sequences from a same set of point clouds that capture an opening Eustoma.*

A side-by-side comparison between our reconstructed mesh and the captured photos is shown in Figure 11. It shows that in all stages of the blooming process the projections of our reconstructed mesh models match very well to the observed 2D petal shapes.

Figure 12 shows a blooming sequence of a Golden Lily, which is particularly complex in early stages (left). The interior petals are hardly visible and hence not well captured in the point cloud. The reconstructed sequence demonstrates our capability to back-track and reconstruct such missing information from very little data while avoiding collisions and preserving petal shapes. Similarly, in Figure 13, the blooming sequence of a Water Lily with long and

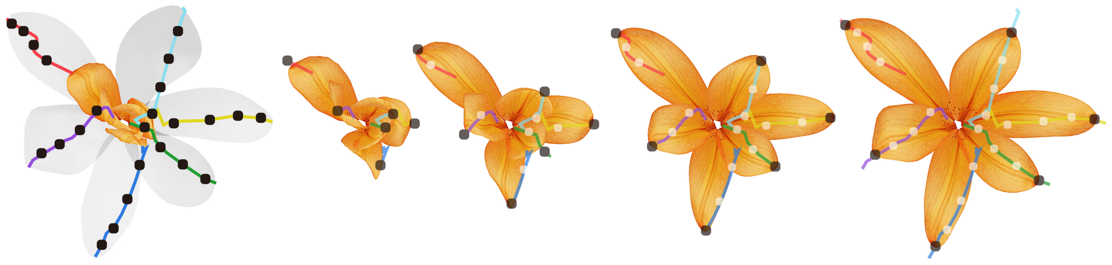

Figure 16: *Motion editing application. Leftmost image shows two overlaying models, where curves represent the trajectories of petal tips. The remaining models generated by interpolating between only two input models.*

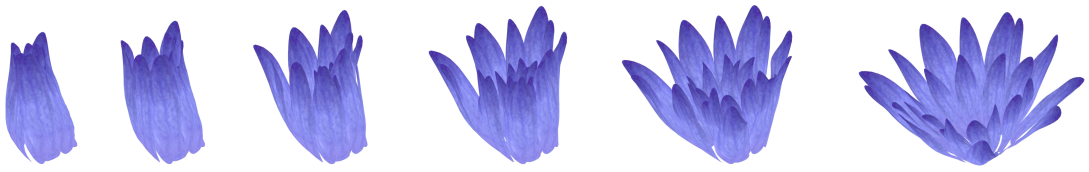

Figure 17: *Animated shape editing. A synthetic two-layer Water Lily is generated using the original data in Figure 12 through copy-and-paste editing of petals and their blooming motions.*

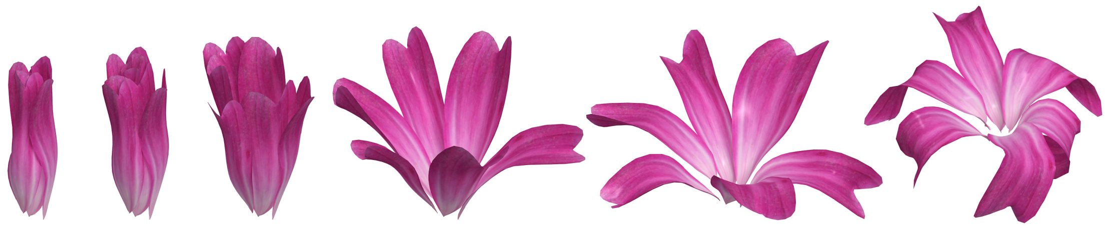

Figure 18: *Motion transfer application. A synthetic yet realistic looking blooming flower of heart-shape petals is generated by transferring the Lily blooming trajectories (Figures 2 and 11) to a set of new petals.*

thin petals is shown. Petals are insufficiently sampled due to their narrow structure. Nevertheless, our template model tracks and fits these petals by enforcing the shape-preservation prior.

To quantitatively evaluate the reconstruction accuracy, we measure the error between the reconstructed surfaces and the raw input data, and monitor how it evolves during the dynamic blooming process. The error metric is defined as the average Euclidean distance between each data point and its closest point on the reconstructed surface. As shown in Figure 14, the fitting error is around one millimeter for all examples, which is quite small compared with the typical size of a flower petal. (A petal is about 50 millimeters long at the early stage and grows longer.) The fitting accuracy of the Lily is relatively lower due to its larger size, but still maintains at the same level as that of the pre-built template.

Influence of template. Figure 15 illustrates the impact of the given petal templates. It is surely hard to clearly identify the number and the shape of individual petals from the captured point clouds of an opening Eustoma. Accordingly, we build two different sets of petal templates by the interactive modeling method, and fit them to the point cloud sequence. Both generated flower blooming sequences are realistic and capture the way how an Eustoma grows and opens; please also see the accompany video. In addition, the fitting errors of both generated sequences, indicated by the yellow and red curves in Figure 14, are quite small and similar.

Applications. Besides outputting a set of models, our approach also provides motion information for different petals, which are represented as a set of transformations. This allows users to easily manipulate flower blooming motion for special effects. As shown in Figure 16, using only two of the reconstructed models and the motion trajectories in between, we can fit the trajectories using B-Spline curves and then resample them at desired non-linear rates. This allows us to adjust the blooming behavior of a flower (e.g. introducing a smooth or sudden blooming motion) or even to adjust the behavior of different petals independently. In both cases new collisions may be introduced. This is addressed by using the interpolated positions as soft constraints and then solving for shape and penetration avoidance to find their optimal positions.

The availability of motion information also allows us to extend the notion of editing and perform copy-and-paste edits for both petal models and petal motions. Specifically, we can copy animated petals and paste them at different places inside a flower and let them bloom there. In Figure 17 a synthesized Water Lily consisting of two petal layers (inner and outer) is animated in a natural way.

Finally, our approach allows to decouple petal shape and motion. The characteristics of blooming motion can be extracted and transferred to another flower. In Figure 18 we show a synthetic but yet realistic blooming flower, generated by transferring the reconstructed blooming trajectories from Lily petals to heart-shape petals.

6. Discussion

In this work we present a novel algorithm for tracking and reconstructing blooming flowers. We capture blooming sequences using commercial off-the-shelf 3D scanners within a simple acquisition setup. The main challenge is recovering parts in space and time that are completely missed by the scanner due to self-occlusions, petal size and overall complexity. In particular at early stages of the blooming process, interior petals can be completely hidden in the data, whereas at later stages, petals usually undergo large deformations of twisting and wrinkling due to decay.

We devise a template-based tracking-and-fitting algorithm, which performs both forward and backward in time. It allows us to recover completely missing geometry by inferring from future steps in the blooming sequence through a constrained optimization technique. Our results demonstrate the ability to reconstruct intricate flower blooming sequences of various species.

Nevertheless, our method is still confined to flowers with relatively simple shapes and configurations. Currently we cannot effectively handle flowers with densely packed petals, such as Rose and Peony. Partly this is because the scan data of these complex flowers are highly incomplete and hence generating a complete template is very challenging even with interactive approaches.

We use skeletons as simple constraints to guide the deformation of flower petals. Some species may not have a simple skeleton structure, but typically a petal has a well-defined central axis and boundaries, which provide sufficient controls to define naturally looking deformations. Figure 8 illustrates our ability to approximate complex petal deformations. Adding branches to the central axis will facilitate approximating even more complex petal structures.

Limitations. A limitation of our method is the level of geometric details that we can reconstruct. While the structured light scanner we used is sufficient for capturing the shape of visible exterior petals, it does not provide sufficient geometric details for modeling the delicate winkle movements. We expect that using high quality laser scanner, or even X-ray for interior petals, will help to ease the problem. Nevertheless, tracking subtle winkle movements adds additional challenges on the algorithm.

Another limitation is temporal coherence, which is not explicitly enforced in our optimization. As a result, when large deformations arise suddenly or the captured input point clouds are vibrating heavily, subtle flickering may occur in the generated animations. Applying a bilateral smoothing filter on the mesh sequence could improve the temporal consistency. It, however, might also filter out subtle blooming developments of a flower.

Future work. We plan to extend our algorithm towards reconstruction of complex botanic species such as bushes, foliage and additional plants. Thus, we intend to generalize our semi-manual technique to reconstruct a range of botanic blooming, growing and deforming phenomena. Additionally, due to the simplicity of our setup, we plan to create a database of various blooming flowers. Our goal here is to analyze the statistics of the space of blooming flowers and obtain novel insights into the process, which can guide the physical-based simulation algorithms. Besides that, space-time dynamic analysis [YLX*16] and reconstruction [WXZ*16] are also very interesting directions to explore.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by NSFC (61522213, 61379090, 61331018), 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015, 2014B050502009, 2014TX01X033, 2016A050503036), Shenzhen Innovation Program (JCYJ20151015151249564), National Foreign 1000 Plan (WQ201344000169), Guangdong Leading Talents Plan (00201509) and NSERC (293127).

References

[BDS*12] Bouaziz S., Deuss M., Schwartzburg Y., Weise T., Pauly M.: Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum (Proc. Eurographics Symp. on Geometry Processing) 31*, 5 (2012), 1657–1667. 7

[BHLW12] Bojsen-Hansen M., Li H., Wojtan C.: Tracking surfaces with evolving topology. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 31*, 4 (2012), 53:1–53:10. 3

[BML*14] Bouaziz S., Martin S., Liu T., Kavan L., Pauly M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 33*, 4 (2014), 154:1–154:11. 7

[CCG*10] Cui M.-L., Copsey L., Green A. A., Bangham J. A., Coen E.: Quantitative control of organ shape by combinatorial gene activity. *PLoS Biol 8*, 11 (2010), e1000538. 3

[CRLM*04] Coen E., Rolland-Lagan A.-G., Matthews M., Bangham J. A., Prusinkiewicz P.: The genetics of geometry. *Proc. National Academy of Sciences 101*, 14 (2004), 4728–4735. 3

[DL10] Deussen O., Lintermann B.: *Digital Design of Nature: Computer Generated Plants and Organics*, 1st ed. Springer Publishing Company, Incorporated, 2010. 2

[DLR77] Dempster A. P., Laird N. M., Rubin D. B.: Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society 39*, 1 (1977), 1–38. 4

[FDM*10] Fernandez R., Das P., Mirabet V., Moscardi E., Traas J., Verdei J.-L., Malandain G., Godin C.: Imaging plant growth in 4d: robust tissue reconstruction and lineaging at cell resolution. *Nature Methods 7*, 5 (2010), 547–553. 3

[GKH*10] Green A. A., Kennaway J. R., Hanna A. I., Bangham J. A., Coen E.: Genetic control of organ shape and tissue polarity. *PLoS Biol 8*, 11 (2010), e1000537. 3

[GWM01] Gumhold S., Wang X., Macleod R.: Feature extraction from point clouds. In *Proc. Int. Meshing Roundtable* (2001), pp. 293–305. 5

[HLZ*09] Huang H., Li D., Zhang H., Ascher U., Cohen-Or D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 28*, 5 (2009), 176:1–176:7. 3

[HWCO*13] HUANG H., WU S., COHEN-OR D., GONG M., ZHANG H., LI G., CHEN B.: l_1-medial skeleton of point cloud. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 32*, 4 (2013), 65:1–65:8. 2

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H.: Edge-aware point set resampling. *ACM Trans. on Graphics 32*, 1 (2013), 9:1–9:12. 3

[IOOI05] IJIRI T., OWADA S., OKABE M., IGARASHI T.: Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 24*, 3 (2005), 720–726. 2

[IYKI08] IJIRI T., YOKOO M., KAWABATA S., IGARASHI T.: Surface-based growth simulation for opening flowers. In *Proc. Graphics Interface* (2008), pp. 227–234. 2, 3

[IYYI14] IJIRI T., YOSHIZAWA S., YOKOTA H., IGARASHI T.: Flower modeling via x-ray computed tomography. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 33*, 4 (2014), 48:1–48:10. 2

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 31*, 4 (2012), 77:1–77:10. 6

[JBPS11] JACOBSON A., BARAN I., POPOVIĆ J., SORKINE O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 30*, 4 (2011), 78:1–78:8. 6

[JPK13] JEONG S., PARK S.-H., KIM C.-H.: Simulation of morphology changes in drying leaves. *Computer Graphics Forum 32*, 1 (2013), 204–215. 3

[KA14] KYRIAZIS N., ARGYROS A.: Scalable 3D tracking of multiple interacting objects. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition* (2014), pp. 3430–3437. 3

[KRP*15] KLEHM O., ROUSSELLE F., PAPAS M., BRADLEY D., HERY C., BICKEL B., JAROSZ W., BEELER T.: Recent advances in facial appearance capture. *Computer Graphics Forum (Proc. of Eurographics) 34*, 2 (2015), 709–733. 3

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Analysis & Machine Intelligence 16*, 2 (1994), 150–162. 7

[LD99] LINTERMANN B., DEUSSEN O.: Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1 (1999), 56–65. 2

[LDS*11] LI C., DEUSSEN O., SONG Y.-Z., WILLIS P., HALL P.: Modeling and generating moving trees from video. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* (2011), 127:1–127:12. 3

[LFM*13] LI Y., FAN X., MITRA N. J., CHAMOVITZ D., COHEN-OR D., CHEN B.: Analyzing growing plants from 4d point cloud data. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 32*, 6 (2013), 157:1–157:10. 1, 3, 4

[LLX*15] LI J., LIU M., XU W., LIANG H., LIU L.: Boundary-dominant flower blooming simulation. *Computer Animation and Virtual Worlds 26*, 3-4 (2015), 433–443. 2, 3, 9

[LM09] LIANG H., MAHADEVAN L.: The shape of a long leaf. *Proc. National Academy of Sciences 106*, 52 (2009), 22049–22054. 3

[LM11] LIANG H., MAHADEVAN L.: Growth, geometry, and mechanics of a blooming lily. *Proc. National Academy of Sciences 108*, 14 (2011), 5516–5521. 3

[MEL*05] MUNDERMANN L., ERASMUS Y., LANE B., COEN E., PRUSINKIEWICZ P.: Quantitative modeling of arabidopsis development. *Plant Physiology 139* (2005), 960–968. 3

[MS10] MYRONENKO A., SONG X.: Point set registration: Coherent point drift. *IEEE Trans. Pattern Analysis & Machine Intelligence 32*, 12 (2010), 2262–2275. 4

[PHM93] PRUSINKIEWICZ P., HAMMEL M. S., MJOLSNESS E.: Animation of plant development. In *Proc. Conf. on Computer Graphics and Interactive Techniques* (1993), pp. 351–360. 2

[PL96] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996. 2

[PNDN12] PIRK S., NIESE T., DEUSSEN O., NEUBERT B.: Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 31*, 6 (2012), 169:1–169:10. 3

[PR12] PRUSINKIEWICZ P., RUNIONS A.: Computational models of plant development and form. *New Phytologist 193*, 3 (2012), 549–569. 3

[PSK*12] PIRK S., STAVA O., KRATT J., SAID M. A. M., NEUBERT B., MĚCH R., BENES B., DEUSSEN O.: Plastic trees: Interactive self-adapting botanical tree models. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 31*, 4 (2012), 50:1–50:10. 3

[QSW*14] QIAN C., SUN X., WEI Y., TANG X., SUN J.: Realtime and robust hand tracking from depth. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition* (2014), pp. 1106–1113. 3

[RLBC03] ROLLAND-LAGAN A.-G., BANGHAM J. A., COEN E.: Growth dynamics underlying petal shape and asymmetry. *Nature 422*, 6928 (2003), 161–163. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Computer Graphics Forum (Proc. Eurographics Symp. on Geometry Processing)* (2007), pp. 109–116. 6

[SLHA13] SCHULMAN J., LEE A., HO J., ABBEEL P.: Tracking deformable objects with point clouds. In *Proc. IEEE Int. Conf. on Robotics & Automation* (2013), pp. 1130–1137. 3, 4

[SNF14] SCHMIDT T., NEWCOMBE R., FOX D.: Dart: Dense articulated real-time tracking. In *Proc. Robotics: Science and Systems* (2014). 3

[TST*15] TAGLIASACCHI A., SCHRÖDER M., TKACH A., BOUAZIZ S., BOTSCH M., PAULY M.: Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (Proc. Eurographics Symp. on Geometry Processing) 34*, 5 (2015), 101–114. 3

[WWY*15] WANG B., WU L., YIN K., ASCHER U., LIU L., HUANG H.: Deformation capture and modeling of soft objects. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 34*, 4 (2015), 94:1–94:12. 3, 4

[WXZ*16] WU B., XU K., ZHOU Y., XIONG Y., HUANG H.: Skeleton-guided 3D shape distance field metamorphosis. *Graphical Models* (2016), 1–9. 11

[XC11] XIAO H., CHEN X.: Modeling and simulation of curled dry leaves. *Soft Matter 7* (2011), 10794–10802. 3

[XYS*16] XIE K., YAN F., SHARF A., DEUSSEN O., CHEN B., HUANG H.: Tree modeling with real tree-parts examples. *IEEE Trans. Visualization & Computer Graphics* (2016). 2

[YGCO*14] YAN F., GONG M., COHEN-OR D., DEUSSEN O., CHEN B.: Flower reconstruction from a single photo. *Computer Graphics Forum (Proc. of Eurographics) 33*, 2 (2014), 439–447. 2

[YHL*16] YIN K., HUANG H., LONG P., GAISSINSKI A., GONG M., SHARF A.: Full 3D plant reconstruction via intrusive acquisition. *Computer Graphics Forum 35*, 1 (2016), 272–284. 2

[YHZ*14] YIN K., HUANG H., ZHANG H., GONG M., COHEN-OR D., CHEN B.: Morfit: Interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 33*, 6 (2014), 202:1–202:12. 4

[YLX*16] YUAN Q., LI G., XU K., CHEN X., HUANG H.: Space-time co-segmentation of articulated point cloud sequences. *Computer Graphics Forum 35*, 2 (2016), 419–429. 11

[ZSZ*14] ZHANG P., SIU K., ZHANG J., LIU C. K., CHAI J.: Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 33*, 6 (2014), 221:1–221:14. 3

[ZYFY14] ZHANG C., YE M., FU B., YANG R.: Data-driven flower petal modeling with botany priors. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition* (2014), pp. 636–643. 2, 4, 6