DOI: 10.1111/cgf.12332
EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)
Volume 33 (2014), Number 2

Flower Reconstruction from a Single Photo

Feilong Yan[1], Minglun Gong[1,2], Daniel Cohen-Or[3], Oliver Deussen[4], Baoquan Chen[5,1]

[1]Shenzhen VisuCA Key Lab / SIAT, [2]Memorial Univ. of Newfoundland, [3]Tel-Aviv Univ. [4]Univ. of Konstanz, [5]Shandong Univ.

Figure 1: *A lily model is reconstructed from a single photo. From left to right: input photo, reconstructed mesh and textured models from the same view direction as the input, rendering result under a different direction.*

Abstract

We present a semi-automatic method for reconstructing flower models from a single photograph. Such reconstruction is challenging since the 3D structure of a flower can appear ambiguous in projection. However, the flower head typically consists of petals embedded in 3D space that share similar shapes and form certain level of regular structure. Our technique employs these assumptions by first fitting a cone and subsequently a surface of revolution to the flower structure and then computing individual petal shapes from their projection in the photo. Flowers with multiple layers of petals are handled through processing different layers separately. Occlusions are dealt with both within and between petal layers. We show that our method allows users to quickly generate a variety of realistic 3D flowers from photographs and to animate an image using the underlying models reconstructed from our method.

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

1. Introduction

For realistic modeling, many outdoor and indoor scenes have to be decorated with high-quality flower models. Obtaining models for flowers in their natural habitat is non-trivial since flower petals are generally fragile with their shapes easily perturbed during the capture. On the other hand, artists are able to create a model for a flower solely based on a single photo. Nevertheless, this is a tedious task that requires experienced users for good results. Our objective is therefore to present a semi-automatic method for general users, which can reconstruct a flower model from a single photograph with little user interaction.

Despite recent advances in acquisition technology, photograph remains a cheap and easy means to capture reality. However, 3D shape reconstruction from a single image is a highly challenging task. The captured shape is ambiguous mainly due to the lack of depth dimension and occlusion. Methods that reconstruct an object from a single photograph usually require some degree of assistance from the user. Moreover, there are always some priors that help reducing the inherent ambiguities in the observed data [HAA97, HEH05, SCN08, JTC09, CZS*13]. Repetitions also provide significant information that can help alleviate ambiguities. Repeated structural elements, like in common architectural models, observed from different directions, offer multi-view geometry application to a single photograph [WFP11].

We exploit the fact that petals from the single flower usu-

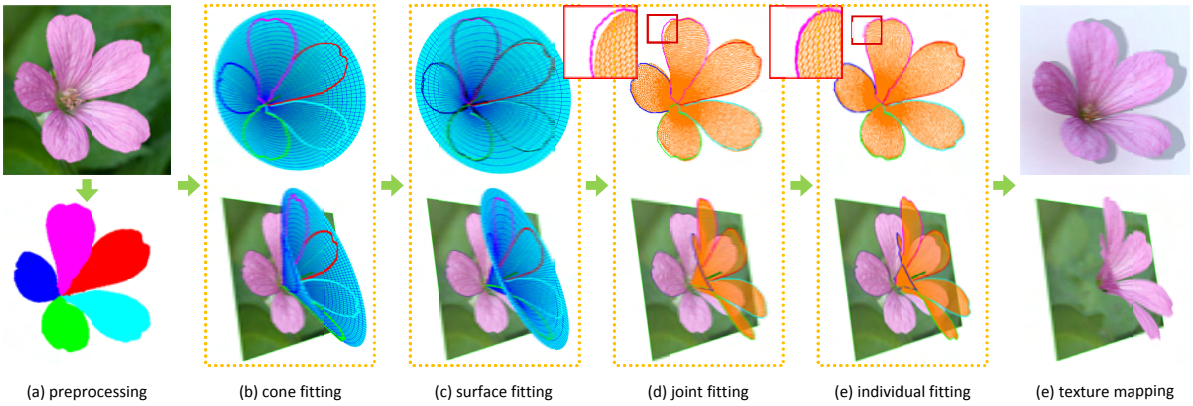


Figure 2: *Flower modeling pipeline: (a) Given a flower photo, we first extract individual petals using available computer vision techniques. (b) The orientation and position of the flower is then estimated through fitting a 3D cone shape. (c) Initialized using the 3D cone, the underlying flower surface is modeled using a surface of revolution and is refined through an iterative procedure. (d) A petal mesh is obtained through trimming the underlying flower surface along a template petal contour and is further deformed to better model the shared features of different petals. (e) In the next step, individual petal mesh is allowed to deviate from the common template to fit each observed petal contour. (f) Finally, textures extracted from the input photo are mapped onto the individual petal meshes.*

ally share similar shapes. A multitude of observations of similar petals within a single image is exploited to reconstruct them. However, unlike man-made objects, flowers impose challenges since i) the repetition of petals is not as structured as it is perceived; ii) no two petals have exactly the same shape, they typically bend in slightly different ways; and iii) petals occlude each other. We combine a global and a local approach to leverage the weak repetition of petals and at the same time to react to the variation of the flower structure within instantiation.

Our method reconstructs flower petals in three main steps. The orientation of the flower is estimated first. A canonical 3D shape is then fitted to the underlying flower structure. Finally, each petal instance is deformed to coincide with the silhouette of individual petal, forming its embedding in 3D. Fig. 1 displays an example of 3D reconstructed flower model and the corresponding input photograph used.

2. Related Work

Most common approaches for plant modeling use a rule-basis to generate a plant from a simple initial state [Hon71, PL90]. Other methods use parameterized algorithms. Starting with Cohen [Coh67], a number of different methods have been developed over the years [dREF*88, LD99], see [DL05] for an overview. Ijiri et al. [IOOI05] present a system for designing flowers, which they later animate [IYKI08].

Other approaches use scanning results, thus having depth information to reconstruct plants. Xu et al. [XGC07] and later Livny et al. [LYO*10] perform this by resorting to global optimization. Livny et al. [LPC*11] combine noisy points of the foliage into what they call lobes, thus avoiding reconstructing smaller details at all.

A number of approaches try to reconstruct plants from photographs. Shlyakhter et al. [SRDT01] extract the visual hull from a set of input images and let a parametric L-System grow into this hull. Reche-Martinez et al. [RMMD04] use a number of precisely registered photographs for the reconstruction of trees. Neubert et al. [NFD07] improve upon this by using a simple flow-based construction mechanism, they only need two or more loosely arranged images. Quan et al. [QTZ*06] and Tan et al. [TZW*07] create a plant model reconstruction from several images using L-Systems; later, they propose a procedural method that generates statistically plausible tree model from a single image [TFX*08].

3. Overview

We observe that most solitary flowers and florets have a single layer of petals. Only a few species, such as lotus and some flowers of capitulum inflorescence, distribute petals in phyllotactic order. Hence in this paper, we first focus on the modeling single-layer flowers and later extend to the flowers with multiple distinguishable layers.

As mentioned above, the key idea of our paper is to exploit the similarity of different petals in the same flower head and the fact that petals are merely foils. Hence, we assume that the 3D shapes of different petals are roughly the same and the observed differences among 2D petal contours are mainly due to the orientation of the petals with respect to the

camera. This makes it possible to reconstruct their 3D geometry. To simplify processing of photos taken by uncalibrated cameras, we further assume that the image of the flower was captured using parallel projection which means the size of the flower is negligible when compared to the distance between the flower and the camera.

With the aforementioned assumptions, our method works as illustrated in Fig. 2. In the preprocessing stage the flower is segmented from its background using GrabCut [RKB04] and each petal is separated using particle flow [NFD07]; see Fig. 2(a). Using the extracted petal shapes, we now estimate the coarse geometry and orientation of the flower by fitting a 3D cone. The tip of the cone is positioned at the flower center and the petals roughly lie on the cone surface (Fig. 2(b)).

Usually the petals of flower heads are bent. Thus, we model the underlying flower surface using a surface of revolution. The estimated 3D cone serves as an initial solution which is gradually refined. For each petal, a contour template is assigned, which describes the typical outline of a petal. The contour templates are projected onto the underlying surface. If the petals would not be bent at all, this would already result in a good solution. By adapting the bending of the surface of revolution we now can improve the fitting of the petal templates (Fig. 2(c)).

Once this fitting process converges, we cut out the petal contours to obtain an initial template mesh for each petal. This mesh is further refined so that its projection fits the petal shape observed in the photo (Fig. 2(d)). While different petals were represented by the same template mesh in the previous joint fitting step, they are now allowed to deviate individually from the common template. Each petal mesh is deformed independently to model the distinctive features of the corresponding petal. The result is shown in Fig. 2(e).

Finally the petals are textured by mapping the corresponding region in the photo onto the surface. For occluded parts without a corresponding texture, texture synthesis is performed.

4. Algorithm

The algorithm starts with extracting the flower head from its background using GrabCut [RKB04]. The center of the flower o is located through user interaction; see Fig. 3(a). Next, the individual petals are located automatically. This is done by traversing the flower contour and measuring distances between points on the contour and the center o. Valleys in this function indicate intersection points between adjacent petals. Starting with each intersection point we trace the petal boundary using the particle-flow method [NFD07]. A particle is traced from the intersection point towards the flower center by following the edge of the petal in the input image (Fig. 3(b)). These traces result in the 2D projective contour c_p for each petal p. Based on this contour we locate the petal tip location t_p as the midpoint on the contour c_p (Fig. 3(c)).

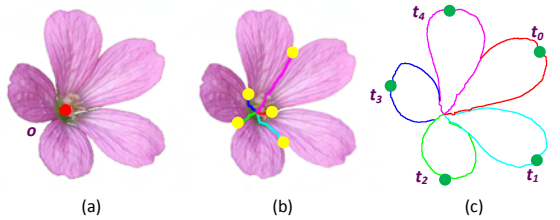


Figure 3: *Preprocessing: Given the extracted flower, we first manually locate the flower center o (red in (a)). The intersection points (yellow in (b)) are then automatically detected, based on which the petals are partitioned using a particle flow method. Finally the tip t_p of each petal p (green in (c)) is automatically located.*

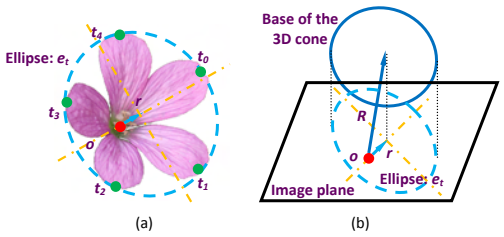


Figure 4: *Cone fitting: An ellipse e_t (blue dashed line in (a)) is fitted to the petal tip points t_p under the constraint that its minor axis (orange dashed line) goes through the flower center o (red dot). This ellipse is the projection of a circle which, together with o, forms a 3D cone (b).*

4.1. Flower Orientation Cone Fitting

Given the extracted petals and detected feature points, we estimate the position and orientation of the flower head by fitting a 3D cone. The apex of the cone is positioned in o, and the base is aligned with the tips of flower petals. Under the assumption of a parallel projection, the projection of the cone's base forms an ellipse whose minor axis direction in the projection goes through o; see Fig. 4(a).

As shown in Fig. 4(b), the ellipse e_t is the projection of the base circle of a 3D cone.Vector **r** connects the flower center o and the center of e_t is the projection of the 3D cone axis **R**, which approximates the axis of the flower. The angle between **R** and **r** affects the ratio of the transverse and conjugate diameters of e_t. Hence, once e_t is obtained, the 3D orientation of the flower **R** is found.

4.2. Underlying Flower Surface Fitting

Now we model the underlying surface that the flower petals lie on. For many flowers this surface can be approximated by a surface of revolution defined by rotating a curve around the main axis of the flower **R**. In fact, the above-defined 3D cone is a special case of such a surface of revolution.

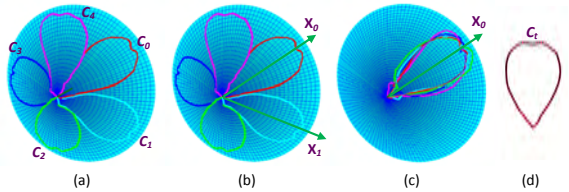


Figure 5: *Computing the template contour: Given the initial cone shape, we back-project each petal p onto the cone to form a closed 2d-curve C_p (a). The main axis $\mathbf{X}_p$ is then computed for each C_p (b). Based on this the different curves are aligned (c). The template petal contour C_t is then computed by averaging (d).*

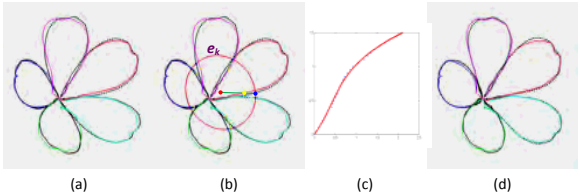


Figure 6: *Fitting of the underlying flower surface: Using the template contour curves (black dashed curves) to represent all petals does not fully capture the shape variation of the individual petals (a). The differences between the two sets of curves are used to guide the fitting of underlying surface (b). Using a point to represent the position and size of each ellipse, a surface of revolution can be computed by fitting a cubic curve to these points (c). Projecting the template contour onto the new surface of revolution results in a better approximation (d).*

Hence, we use the cone as an initial surface and iteratively refine the rotating curve. The refinement is based on the aforementioned assumption that different petals have similar geometric shape, i.e., the observed differences among petal contours c_p are caused by the bending of the underlying surface. Hence, the projections of the same template contour plotted at different locations of the underlying surface should closely approximate c_p for different petals p.

Computing the template petal contour. Our first goal is to compute a template petal contour C_t on the surface of revolution that captures the average shape of the different petals. This is achieved by back-projecting the 2D contour c_p of each petal p onto the surface of revolution (Fig. 5(a)). The projection of each contour c_p forms a closed curve C_p in 2d on the surface of revolution. We further compute the main axis $\mathbf{X}_p$ for each curve C_p, which divides the area into two equal parts; (Fig. 5(b)). All the curves are aligned using their main axes to obtain an average petal contour. Once aligned, the averaging provides us with what we call the template petal contour C_t (Fig. 5(d)).

Update underlying surface. Once the template contour C_t is obtained, we can use it to replace all individual curves C_p on the surface of revolution. Since C_t does not fully capture the shape variation of individual petals, the projections of the template contour on the image plane do not follow the individual petal contours c_p very well (Fig. 6(a)). In the next step we use the differences between the two curve sets to adjust the underlying surface.

To illustrate surface fitting, we treat the surface of revolution as a set of 3D circles whose centers lie on the axis of the flower $\mathbf{R}$. The projection of each 3D circle k on the image plane forms an ellipse e_k, whose center lies on the vector $\mathbf{r}$. If we alter the radius of k, the projection of the template contour on the surface of revolution would move either towards or away from the center of e_k. Based on this observation we adjust the radii of the 3D circles so that the overall distance between the projections of the template contours and the real petal contours is minimized.

The process is shown in Fig. 6(b): to adjust the radius of a given 3D circle k, we first locate points (in blue) along the template contour that lie on the ellipse e_k (in red). We then connect these points with the center of the ellipse (shown as the red point) and find their intersections (in yellow) with the real petal contours. Finally, we scale the current 3D circle so that the sum of squared distances between the yellow points and the newly projected ellipse e'_k is minimized. After the new radii for different 3D circles are calculated, we fit a cubic curve to them (Fig. 6(c)). This defines the new surface of revolution. Template contour generation and surface fitting are repeated in alternation until convergence.

4.3. Joint Petal Mesh Fitting

Now we convert the template contour defined on the flower surface into a 3D template petal mesh, which is further deformed to better fit the observed petal contours. Converting the contour into a mesh is rather trivial. We first sample the area enclosed by the contour using a set of points on the underlying surface of revolution. These points, together with sampled points on the contour, are then used to construct a triangle mesh.

The template petal mesh obtained this way lies on the underlying flower surface. However, a petal in real world is often bent individually. Hence, the template meshes need to be deformed to better approximate the shape of the petals. The deformation is again guided by the observed petal contours and based on the assumption that different petals have similar geometry. To be more precise, the deformation tries to drag the vertices on the boundary of the template mesh so that the differences between the observed contours and the projections of the corresponding template meshes are minimized.

To facilitate the deformation, we introduce a local coor-

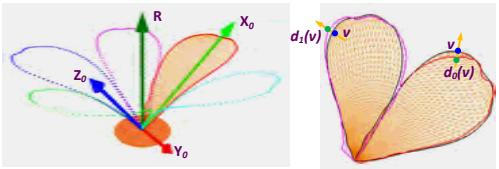


Figure 7: *Joint petal mesh fitting: (a) local coordinate system defined for a petal mesh; (b) locating the target points (green) for the same given vertex v (blue) on the boundary of the template mesh.*

dinate system for each template mesh p along its main axis $\mathbf{X}_p$. The origin is located at the flower center o. The $\mathbf{Y}_p$ axis points along the cross product $\mathbf{X}_p \times \mathbf{R}$, and $\mathbf{Z}_p$ along $\mathbf{Y}_p \times \mathbf{X}_p$ (Fig. 7(a)).

Let vector $\mathbf{l}_v$ denote the local coordinates for a given vertex v on the boundary of the template mesh; and let matrix $\mathbf{M}_p$ project the template mesh to the location of petal p on the image plane. We now move vertex v to a new location $\mathbf{l}'_v$ so that the projections $\mathbf{M}_p\mathbf{l}'_v$ are close to the petal contour c_p for all petals p. To achieve this goal, we first compute a target point $d_p(v)$ for vertex v on each c_p. This is done by first estimating the tangent of the projected template contour at location $\mathbf{M}_p\mathbf{l}_v$, and then searching the nearest intersection between c_p and the 2D plane perpendicular to the tangent. With the target points of v found on all contours, the location L'_v is computed through minimizing the sum of squared distances between the projections of the vertex v and the corresponding target points, i.e., $\sum_p \|\mathbf{M}_p\mathbf{l}'_v - d_p(v)\|^2$.

Once the new locations $\mathbf{l}'_v$ for all boundary vertices v are found, Laplacian smoothing is applied to remove high frequency noises. Then, the template mesh is deformed using mean value geometry encoding [KS06], which preserves the original geometric features of the surfaces. That is, the problem is formulated as minimizing:

$$Energy = w_{geo} * E_{geo} + w_{con} * E_{con} \qquad (1)$$

where E_{geo} is the geometric preserving energy that maintains the geometric features of the mesh; and E_{con} is the contour fitting energy that drags boundary vertices to their new locations. w_{geo} and w_{con} are regulation parameters, which are set to 1.0 and 0.02 for all our examples, respectively.

The deformation is iteratively performed, with the projections of the template mesh gradually fitting into the observed contours. Since multiple contours jointly control the same template, the resulting mesh models the shared geometric features of the flower petals.

4.4. Individual Petal Mesh Fitting

The joint fitting does not model the shape variations of the individual petals. Hence, in the last step, different copies of the template petal mesh are deformed independently based on their petal contours. The deformation process is similar to the one discussed above, but uses a different way to calculate the locations for boundary vertices on the mesh. That is, for a given vertex v on the boundary of the mesh for petal p, we first locate its target point $T_p(v)$ using the same method as described above. The new location L'_v for vertex v is computed as the closest 3D point to L_v that projects to $T_p(v)$. After the new locations for all vertices on the boundary are calculated, the mesh for petal p is updated using Eq. (1).

4.5. Occlusion Handling

The aforementioned baseline algorithm does not consider occlusions among different petals and works only for inputs with limited occlusion, such as shown in Fig. 2. In reality, however, petals often overlap with each other. To handle these cases properly, the following strategies are applied.

Occlusion detection. When two adjacent petals p, q occlude each other, at the intersection point of their contours c_p and c_q, two curves are merged into one. Three tangent directions can be calculated close to such a point: the tangent directions $\mathbf{T}_p$ and $\mathbf{T}_q$ for contours c_p and c_q (before merging) and tangent direction $\mathbf{T}'$ for the merged contour (Fig. 8(b)). If we assume that the unoccluded petal contours are locally smooth, the tangent direction for the visible contour should not change much before and after the intersection point. Hence, our rationale is that the petal p with dot product $\mathbf{T}_p \cdot \mathbf{T}'$ closer to -1 (smoother continuation) is the one that is visible. Once the occlusion relationship is determined, we assign a weight to each point on the petal contour. Visible points have weights of 1.0, whereas the weights of occluded ones gradually drop to 0 based on their distances to the closest visible point.

Occlusion-aware petal fitting. With different contour points having different weights, the petal fitting process is adjusted so that occluded contour points have less influence to the result than visible ones. When we compute the template petal contour C_t, the weighted average is used instead of the simple average so that the obtained shape is less affected by the occluded points. Furthermore, during surface fitting, occluded contour points have less influence when fitting new ellipses e'_k for obtaining the surface of revolution (see Sec. 4.3). Similarly in the joint fitting step, the weight of each target point $d_p(v)$ on the contour is also taken into account.

The situation is a bit different for the individual fitting step since the optimal solution is no longer over-determined. Hence, using weights to control the deformation cannot fully eliminate the influences of occluded parts. To address this problem, we deform the mesh using visible points only and use Laplacian smoothing to fill in the occluded parts. This leads to reasonably consistent petal meshes.

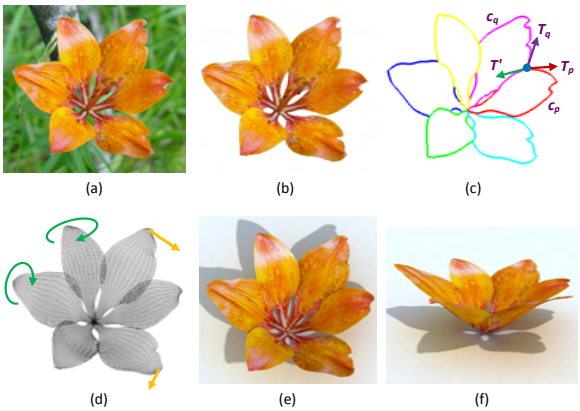


Figure 8: *Occlusion handling: (a) input photo; (b) extracted flower; (c) three tangent directions ($\mathbf{T}_p$, $\mathbf{T}_q$, and $\mathbf{T}'$) estimated at a given intersection point (blue) are used to determine the occlusion relationship; (d) based on the occlusion relations, two petals are dragged away from the camera (along the orange arrows) and two petals are rotated about their main axis (along green arrows); (e) the final model with texture map; (f) another view of the model.*

Finally, the occlusion relationship also allows us to infer the relative positions of adjacent petals in 3D space, i.e., a visible petal is closer to the camera than its occluded neighbor. If a petal p is occluded by its two adjacent neighbors, we add an additional occlusion handling energy E_{occ} to drag its tip t_p away from the camera during individual petal fitting. If petal p is occluded from one side but visible on the other, the energy E_{occ} will be used to rotate p about $\mathbf{R}_p$ so that the visible side is closer to the camera and the occluded side is further away; see Fig. 8(c). E_{occ} is incorporated into Eq. 1 with the corresponding regulation parameter w_{occ} set to 0.02. Since it works with geometric preserving energy E_{geo} and contour fitting energy E_{con}, the occlusion corrected petal meshes will maintain geometric features of the template petal mesh and respect the observed contours.

4.6. Extension for Multi-Layer Flowers

Another direction for enhancing the baseline algorithm is to handle multi-layer flowers. Here we assume petals of these flowers can be roughly grouped into a few layers. This allows us to simply apply the baseline algorithm for modeling petals on different layer separately and then combine the obtained petal meshes into the same model. Nevertheless, as shown in Fig. 9(a), the contours of petals belonging to inner/upper layers do not show up on the silhouette of the flower. Hence, the aforementioned petal segmentation approaches based on valleys on silhouette won't work. A different preprocessing process is applied to semi-manually detect and segment petals, as well as grouping them into layers.

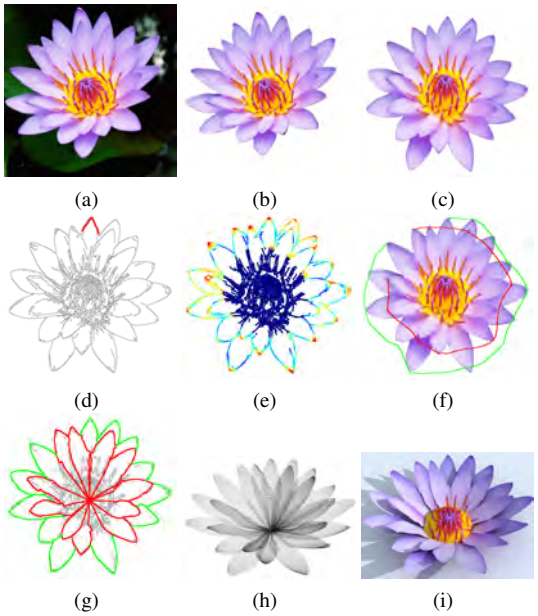


Figure 9: *Petal detection and grouping for multilayer flower: (a) input photo; (b) extracted flower; (c) reprojected to front-parallel view; (d) edge pixels detected with petal template drawn by the user shown in red; (e) result of Chamfer matching where warmer colors correspond to high matching scores; (f) user scribbles for assigning petals into layers; (g) petal segmentation result; (h) reconstructed mesh model; (i) textured model.*

The process also starts with the user identifying the flower center o. The cone fitting step is then performed using o and points on the flower silhouette. Based on the estimated 3D cone, we reproject the flower to the front parallel view so that the variation on petal shapes due to the projection is minimized; see Fig. 9(b). Next, we detect edge pixels using the reprojected photo and ask the user to draw the contour for one of the petals; see Fig. 9(c). This contour serves as a template for detecting the tip locations of the remaining petals using Chamfer matching [TLO10]. A 2D radial searching space θ, d is used for Chamfer matching, where θ determines the orientation of the template and d is the distance between the tip of the template and the center o. That is, we rotate the template about the flower center o and translate it to different distance to the center o, before measuring the matching scores. Fig. 9(d) demonstrate the result of Chamfer matching, where high score pixels correspond the locations of the detected petal tips.

Once the tip locations of different petals are detected, the user is required to group petals into layers by drawing scribbles around them; see Fig. 9(e). The abovementioned particle flow method is then applied to segment the petals layer by layer, starting from the top one. For the top layer, we trace

the particles from the tip of the petal until they reach the flower center o. For the bottom layers, we stop the particles once they reach the contours of the petals in previous layers. The results of petal segmentation are shown in Fig. 9(f), which are used for modeling the petal meshes layer by layer.

5. Results

Our algorithm is evaluated using a variety of flower photos downloaded from the Internet. On average, preprocessing takes less than half a minute. Our automatic reconstruction algorithm, even for the most complex cases, takes less than a minute. To the best of our knowledge, experienced artists often find it tedious and time consuming to create a flower model that resembles the appearance of a given photo. They need half an hour or more to create a highly-detailed flower model. In contrast, our algorithm offers much simpler and faster way to model flowers and can be quickly mastered by users without any 3D modeling skills.

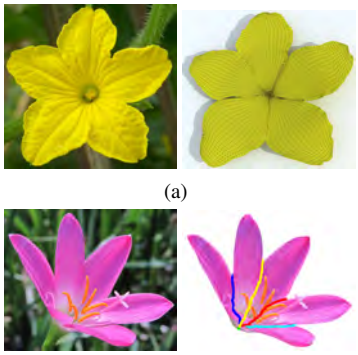
The flowers in Figs. 1, 11(b), 11(f), and 11(i) contain highly curved petals. Our approach properly models the underlying geometry using surfaces of revolution, regardless of whether the petals bend toward the flower center (Fig. 11(b)) or away from it (Figs. 11(f), 11(i)). The flower petals in Figs. 11(c) and 11(d) have asymmetric shapes and strong shape variations, which are properly modeled through the joint and individual petal fitting steps. The ones shown in Figs. 11(a), 11(b), and 11(e) have complex occlusions. Our approach detects the correct occlusion relationships in most cases and deforms the petal meshes accordingly.

Figs. 11(g) and 11(h) show two more flowers with multiple layers of petals. Since the layers lie on different surfaces, directly applying our techniques on all petals would lead to unrealistic models. Hence, the process discussed in Section 4.6 is applied to semi-manually segment the petals and group them into different layers.

Figs. 11(i) and 11(j) demonstrate that our approach can estimate coherent flower models for multiple flowers of the same type. Due to the variation of the flower shapes and view directions, modeling individual flowers separately would yield models with quite different geometry. This problem is addressed by fitting the same underlying geometry to the surfaces and deforming the same template petal mesh in the joint fitting step. The template petal mesh obtained is subsequently used to fit petals from individual flowers. The resulting flower models share similar geometry, but yet capture the shape variations of different flowers.

Finally, the accompanying videos show that an input photo with multiple flowers can be easily animated using reconstructed flower models; see Fig. 11(k) for two of the frames.

Limitations and Failure Cases. Modeling flowers using single photos has its inherent limitations. First, the proposed approach estimates the petal geometry based on the observed



(a)

(b)

Figure 10: *Examples for limitations: (a) the rugged surface on the a cucumber flower cannot be properly modeled; (b) when the center of the zephyr lily is projected outside of the ellipse formed by petal tips, the particle flow method fails to partition the petals.*

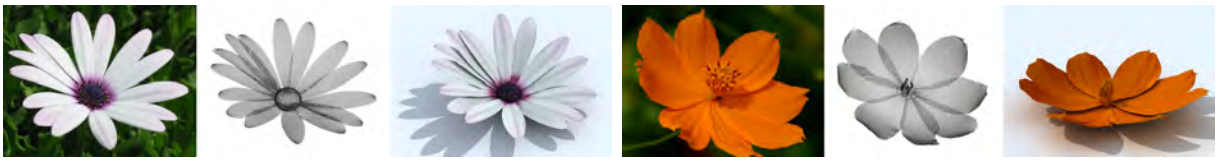
petal contours only. It can generate individual petal meshes that accurately follow the observed contours, but cannot fully recover fine geometry details (lumps and dents) on the surface. Recovering these details from a single photo under uncontrolled illumination would be very difficult; see Fig. 10(a).

Secondly, the particle flow based petal segmentation implicitly assumes that the flower center o is located inside of the envelope formed by the tips of the petals, which puts constraints on the usability of the input photos, i.e., the photo cannot be taken from a highly oblique view directions; see Fig. 10(b). On the other hand, when the input photo is taken directly above the flower, the observed contours for different petals are similar. This makes it difficult to infer the 3D shape of the underlying surface, leading to the reconstructed petal meshes being unnaturally flat; see Fig. 11(e) and 11(g).

Finally, our occlusion detection approach decides based on the normals of the two contours at the intersection points; see Fig. 8. It works well for general cases, but does fail sometime. For example, in Fig. 11(a), the occlusion relation between two petals along the top left direction is incorrectly detected. For multilayer flowers, the Chamfer matching method that we employed may fail to detect all the petal tips; see Fig. 9(f) where a petal along the top left direction is missed.

6. Conclusion and Future Work

In this paper we present a method for capturing the geometry of flowers from a single input photograph. We take advantage of the fact that many similar petals are seen from slightly different orientations. This allows us to gain additional knowledge about the underlying surfaces. In subsequent steps we firstly fit a cone, later a surface of revolution, and



(a) A daisy flower with occlusion relations among petals. (b) Another daisy with petals bend toward the main axis of the flower.

(c) A nerium oleander with asymmetric petal contours. (d) A common mallow with strong shape variation among petals.

(e) A gazania rigens viewed from the top, resulting in a rather flat mesh. (f) Another gazania rigens viewed from the side.

(g) A double balloon with 2 layers (color coded in mesh) of petals. (h) A water lily with 3 layers (color coded in mesh) of petals.

(i) Two similar plumeria flowers in the same photo. (j) A cluster of lilies in the same photo. Left: input photo; Right: reconstructed mesh.

(k) A cluster of sunflowers. Left: input photo; Middle and right: two frames from a manually created animation in which several reconstructed sunflowers in the foreground follow the sun. The animation is submitted with the paper.

Figure 11: *A variety of flowers modeled using the proposed technique. The left side in (a-i) shows the input photos, middle shows the reconstructed mesh, and right the rendering result.*

finally individual surfaces to the petals and combine them to a 3D model. We developed methods for occlusion detection and removal and demonstrate the quality of our results in a number of examples.

Being an initial attempt of modeling flowers from single images, the proposed approach concentrates on rather simple flowers that allow us to detect and fit the petals using surfaces of revolution. There are many other types of flowers exist that include more complex arrangements of petals; prominent examples are roses or moms. For such cases we would like to extend our method to incorporate other petal layout priors. Furthermore, our current approach does not explicitly model the imperfections of flower heads. If a petal is broken or buckled, the modeling process may not work properly since the assumption on petal similarity does not hold. We plan to automatically detect imperfect petals in the future. Our approach has also some limitations to faithfully model the shapes for flowers with severe occlusions and significant petal deformations (see Fig. 11(k)). Here the user has to assist the system for finding the hidden petal contour.

In summary, our work is a first step in the automatic 3D modeling of flowers from photographs. We show that how the problem can be tackled with proper priors and minimal user interaction. We believe that the proposed approach can be extended in various ways and it may inspire other more complex modeling techniques.

Acknowledgements

The authors would like to thank all the reviewers for their valuable comments. This work was partially supported by grants from NSFC (61379090, 61232011, 61272327), Guangdong Science and Technology Program (2011B050200007), Shenzhen Innovation Program (CXB201104220029A, KQCX20120807104901791, JCYJ20130401170306810,ZD201111080115A, KC2012JSJS0019A), CAS Visiting Professorship for Senior International Scientists, Natural Science and Engineering Research Council of Canada (293127) and the Israel Science Foundation.

References

[Coh67] COHEN D.: Computer simulation of biological pattern generation processes. *Nature*, 216 (1967), 246–248. 2

[CZS*13] CHEN T., ZHU Z., SHAMIR A., HU S.-M., COHEN-OR D.: 3-sweep: extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 195. 1

[DL05] DEUSSEN O., LINTERMANN B.: *Digital Design of Nature: Computer Generated Plants and Organics*. 2005. 2

[dREF*88] DE REFFYE P., EDELIN C., FRANÇON J., JAEGER M., PUECH C.: Plant models faithful to botanical structure and development. In *SIGGRAPH '88* (1988), pp. 151–158. 2

[HAA97] HORRY Y., ANJYO K.-I., ARAI K.: Tour into the picture: using a spidery mesh interface to make animation from a single image. SIGGRAPH '97, pp. 225–232. 1

[HEH05] HOIEM D., EFROS A. A., HEBERT M.: Automatic photo pop-up. *ACM TOG 24*, 3 (July 2005), 577–584. 1

[Hon71] HONDA H.: Description of the form of trees by the parameters of the tree-like body. *Theoretical Biology 31* (1971), 331–338. 2

[IOOI05] IJIRI T., OWADA S., OKABE M., IGARASHI T.: Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. In *SIGGRAPH '05* (2005), pp. 720–726. 2

[IYKI08] IJIRI T., YOKOO M., KAWABATA S., IGARASHI T.: Surface-based growth simulation for opening flowers. In *GI* (Toronto, Ont., Canada, Canada, 2008), pp. 227–234. 2

[JTC09] JIANG N., TAN P., CHEONG L.-F.: Symmetric architecture modeling with a single image. *ACM TOG 28*, 5 (Dec. 2009), 113:1–113:8.

[KS06] KRAEVOY V., SHEFFER A.: Mean-value geometry encoding. *International Journal of Shape Modeling 12*, 1 (2006), 29–46. 5

[LD99] LINTERMANN B., DEUSSEN O.: Interactive modeling of plants. *IEEE CG&A 19*, 1 (1999), 56–65. 2

[LPC*11] LIVNY Y., PIRK S., CHENG Z., YAN F., DEUSSEN O., COHEN-OR D., CHEN B.: Texture-lobes for tree modelling. *ACM TOG 30*, 4 (July 2011), 53:1–53:10. 2

[LYO*10] LIVNY Y., YAN F., OLSON M., CHEN B., ZHANG H., EL-SANA J.: Automatic reconstruction of tree skeletal structures from point clouds. *ACM TOG 29*, 6 (Dec. 2010), 151:1–151:8. 2

[NFD07] NEUBERT B., FRANKEN T., DEUSSEN O.: Approximate image-based tree-modeling using particle flows. *ACM TOG 26*, 3 (2007), Article 71, 8 pages. 2, 3

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. 1990. 2

[QTZ*06] QUAN L., TAN P., ZENG G., YUAN L., WANG J., KANG S. B.: Image-based plant modeling. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 599–604. 2

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": interactive foreground extraction using iterated graph cuts. *ACM TOG 23*, 3 (Aug. 2004), 309–314. 3

[RMMD04] RECHE-MARTINEZ A., MARTIN I., DRETTAKIS G.: Volumetric reconstruction and interactive rendering of trees from photographs. *ACM TOG 23*, 3 (2004), 720–727. 2

[SCN08] SAXENA A., CHUNG S. H., NG A. Y.: 3d depth reconstruction from a single still image. *IJCV 76*, 1 (2008), 53–69. 1

[SRDT01] SHLYAKHTER I., ROZENOER M., DORSEY J., TELLER S.: Reconstructing 3d tree models from instrumented photographs. *IEEE Comput. Graph. 21*, 3 (2001), 53–61. 2

[TFX*08] TAN P., FANG T., XIAO J., ZHAO P., QUAN L.: Single image tree modeling. *ACM TOG 27*, 5 (2008), 108. 2

[TLO10] THANH N. D., LI W., OGUNBONA P.: An improved template matching method for object detection. In *Computer Vision–ACCV 2009*. Springer, 2010, pp. 193–202. 6

[TZW*07] TAN P., ZENG G., WANG J., KANG S. B., QUAN L.: Image-based tree modeling. *ACM TOG* (2007), Article 87. 2

[WFP11] WU C., FRAHM J.-M., POLLEFEYS M.: Repetition-based dense single-view reconstruction. In *CVPR* (2011), pp. 3113–3120. 1

[XGC07] XU H., GOSSETT N., CHEN B.: Knowledge and heuristic-based modeling of laser-scanned trees. *ACM TOG 26*, 4 (Oct. 2007). 2