DOI: 10.1111/cgf.13557
Pacific Graphics 2018
H. Fu, A. Ghosh, and J. Kopf
(Guest Editors)

Volume 37 (2018), Number 7

Reconstructing Flowers from Sketches

Cédric Bobenrieth[1], Hyewon Seo[1], Frédéric Cordier[2] and Arash Habibi[1]

[1] CNRS, University of Strasbourg, ICube, France
[2] University of Haute-Alsace, LMIA, France

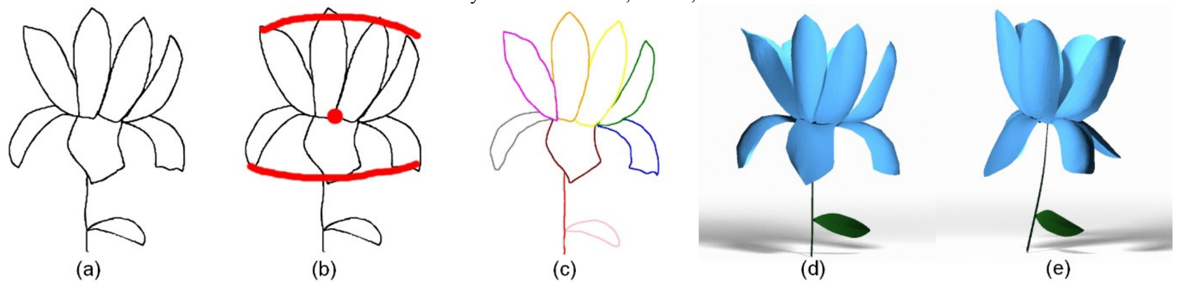


Figure 1: *Reconstruction of a flower model from a sketch: the input sketch (a), the guide strokes provided by the user which will be used for the 3D cone reconstruction (b), the segmented sketch into petals and other botanical elements (c), the reconstructed model from the same point of view as the input (d); and from a different view (e).*

Abstract
As the symbol of beauty, floral objects have been one of the most popular subjects of artistic drawing. However, designing 3D floral models is generally time- and resource-consuming, because of their structural and geometrical complexity. In this paper, we address the problem of reconstructing floral objects from sketch input. The user draws a relatively clean sketch of a flower and a few additional guide markings from an arbitrary view to rapidly create quality geometric models of flowers. Our system offers a novel modeling scheme compared to several existing flower modelers accepting sketch as input, where the user is required to work with different views, providing step-by-step sketch input. Given the silhouette and the guide strokes, an assumed, common botanical structure is estimated, i.e. a cone for each ring of petals. The cones and the silhouette sketch that we segment into elementary curves are used to retrieve model elements from the pre-constructed shape database. These elements are then placed together around the cone, where an additional, per-element deformation is performed so as to maximize the silhouette similarity between the user sketch and the 3D flower model from the chosen view. Our system has shown to robustly create a variety of flowers in various configurations, including flower models with several petal layers and various blooming degrees, drawn from different views.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

1. Introduction

Flowers have been the subject of art since the beginning of human art history. However, geometric modeling of flowers can still be very tedious because of their structural and geometrical complexity; flowers are usually composed of a large number of elements and most of them have very thin and elongated shapes which require careful inter-placement from each other. Therefore, an experienced user must carefully work on both the hierarchical structure and fine details when modeling floral objects. In addition, flowers exhibit a large variety of shapes and colors, making it difficult to adapt data-driven approaches that have already been successfully applied to human faces [BV99] or bodies [ACP03, SM04].

Traditionally, floral objects have been modeled by using procedural modeling, such as the L-system. Although such rule-based systems guarantee the creation of models with botanically faithful structure, they leave little room for creative design. Recently, several attempts have been made, proposing alternatives to the engineering-oriented modeling process. Recent shape capture techniques have shown to offer semi- or fully automatic reconstruction solutions for floral [YGC*14, IYYI14, ZYFY14] and botanical objects [QTZ*06, YHG*16], by using real world flower data such as 2D or 3D images. In parallel, interaction techniques devoted to modeling floral objects have been developed, which provide a designer-driven solution to the problem [IOOI05, IOI06].In this paper, we present a new system for reconstructing floral objects from a user-drawn 2D sketch input. To our knowledge, we are the first to address a 3D modeling system for floral objects from a single-view sketch of arbitrary viewpoint. Our work is based on the observation that botanical plausibility is often simplified in a designer-driven modeling of floral objects, as is the case with our perception of them — The perception as well as the drawing of floral objects often simplifies geometric details and repeated arrangement of petals. Some parts (typically the calyx) is sometimes completely omitted. We also benefit from the fact that the floral objects share a common structure which is a relatively simple geometric form (i.e. cone [YGC*14]). With this in mind, we achieve reliable flower shape results by making use of a shape database and a sketch based shape retrieval followed by a non-rigid deformation.

Our method has several advantages over existing modelers for the floral objects. Firstly, it enables the user to rapidly create flowers from an arbitrary view and with minimal interaction, requiring only the silhouette sketch together with a few guide strokes. Secondly, based on a pre-constructed model database, it can create sophisticated flower objects from simplified drawings. In particular, our system can be used to generate flower models with several layers of petals and even flowers with different petal shapes or with petals folding on themselves.

2. Related Work

2.1 Modelers for floral objects

Modelers devoted to floral objects have been developed along two complementary approaches. Interactive systems allow one to model imaginary, artistically interpreted or perceptually simplified flowers. Reconstruction methods, on the other hand, allow to model only those that can be observed. While our work belongs to the interactive approaches, it shares some technical aspects of the reconstruction methods, due to the fact that the sketch is drawn in a single view, which usually contains occluded parts.

Interactive systems. Ijiri et al. [IOI06] have developed an interactive modeling system for designing a scene consisting of multiple flowers in a top-down manner, i.e. from an initial sketch, specifying the overall appearance, to fine details of the desired final model. As in this work, our approach makes use of a model library but we use the sketch input directly for the automatic selection of the model elements whereas in their system, it is the user who performs selection.

Ijiri et al. also [IOOI05] developed an interactive flower modeler that offers a two stage modeling for biologically plausible flower models. The user first designs the structure of a flower model, guided by the floral diagrams and inflorescences [DC10]. The geometry of flower elements as well as an inflorescence is then created by using 2D freeform strokes in the geometry editors.

Reconstruction methods. Recently, Yan et al. [YGC*14] showed how to reconstruct textured flower models from a single photo input. They make use of prior knowledge on the flower structure and redundant appearance of petals to attempt to reconstruct the flower including occluded parts. However, their results are restricted to widely open flowers with nearly front-facing views, as their technique assumes total or partial visibility of each petal. In our work, users have a large freedom for the view selection in their sketch, although they tend to choose frontal views in most cases. Zhang et al. [ZYFY14] have adopted a data-driven approach to the acquisition of flower geometry. A morphable petal shape model has been built from scanning individual flower petals, prior to individual flower modeling from a 3D point cloud scanned from a single view. Ijiri et al. [IYYI14] have adopted X-ray CT (computed tomography) of real flowers for the realistic modeling of flowers. Relying on the volume data for the flower shape including the occluded part, they combine interactive and variational techniques to fit shaft and sheet primitives to target volume regions.

2.2 Assembly based modeling with sketch input

Composing existing models to create new models or scenes has become a method of choice, when aiming at rapid prototyping of objects with several different parts [LF08, SAG*13,XXM*13] or complex scenes containing multiple objects [SI07, XCF*13]. One of the earliest works in assembly-based modeling is the modeling-by-example system proposed by Funkhouser et al. [FKS*04]. The user constructs a new 3D model by selecting a part of the current model to replace, searching for similar parts in the database, and further editing the chosen part model so as to combine it with the current model.

Closer to our work are the recent assembly-based modelers that support sketch input. In Shin and Igarashi [SI07], each time the sketch of a new object in the scene is drawn by the user, the system searches in a database for 3D models whose 2D projections are similar to the sketch. The user can then select the intended model among them and position it in the scene. Lee and Funkhouser [LF08] propose a method enabling the user to create objects or scenes by drawing the sketch step by step. Similarly, Shtof et al. [SAG*13] present an interactive system, where the user drags and drops a series of selected 3D primitives over the sketch input. An optimisation procedure matches and appropriately locates the user-selected 3D primitives to the intended part of the sketch. These methods, unlike ours, require the user to provide drawings interactively, one part at a time. This constraint allows the user to adjust the orientation of the current 3D model so as to be able to draw the part to be added in its entirety. In a similar spirit Xie et al. [XXM*13] propose a method where the user, after having selected an initial 3D model, can easily change a part of it by drawing the desired shape over the actual 3D part. Here again, the system is based on a strong interaction with the user. Moreover, the temporal and spatial context of the sketch input provides information on the type of the 3D part to be reconstructed. In such settings, the system does not need to handle potential occlusions in the drawing.

Our work shares the similar idea of using existing models in the database. On the other hand, ours requires minimal interactions with the user and automatically selects the 3D models that will be used for the reconstruction of a flower. Unlike existing methods, our approach takes as input the complete user sketch with possible occlusions. It does not require any complementary information of the type of each part present on the sketch or its arrangement. It only assumes that the whole drawing currently represents a flower.A more sophisticated form of the model-based approach has been presented by Xu et al. [XCF*13], who make use of both models and their relative configurations, in the indoor scene reconstruction setting. In the preprocessing, a learning phase analyzes a large set of 3D scenes and creates "structural groups", consisting of the presence of frequent relations between objects, as well as a set of possible positions with respect to each other: A television is placed at the center of a table most of the time. Then when the system takes a sketch as input, it detects all objects in it and places 3D models in the database according to the structural groups. In our work, we also make use of structural semantics that are specific to the flower objects. However, our system does not require learning but rather computes the model placement directly from the user input, either in the form of data- or guide-sketch.

3. Overview

Our system is a sketch-based 3D modeling tool for floral objects. The user draws lines depicting the silhouette of a flower seen from an arbitrary view, in order to generate a 3D mesh corresponding to the 2D sketch. We assume a moderate level of abstraction in the input silhouette sketch; overly abstracted drawings, or highly detailed drawings with shading strokes as found in scientific illustration are not handled well in our current system. As shown in Fig. 1, a typical sketch of a flower is composed of petals whose bases are co-located around the flower center. Our current system assumes only one flower is drawn in the sketch. Extending it to simultaneously reconstruct multiple flowers would be trivial, provided that a limited degree of occlusion is exhibited among flowers in the sketch.

Our system first estimates the parameters of one or more *3D cones* representing the global geometry of the corolla. To do so, the user has to provide additional *guide strokes* on the sketch: *a flower center* (i.e. the junction point between the stem and the corolla) and *petal-tip curves* connecting the tips of drawn petals, one for each petal layer. For each set of guide strokes, a 3D cone is constructed (Section 6), which will be used as the layout base for the spatial arrangement of floral elements. The system then analyzes the input sketch strokes based on the 3D cone as well as some floral prior, so as to segment them into a set of elementary curves, each corresponding to a botanic element, such as petal, leaf, and the flower center (comprising the stamen and floral receptacle) (Section 5). The segmented curves are used as query input to retrieve 3D elementary floral shapes in the pre-constructed database (Section 4). Finally, the 3D shape of the flower is reconstructed by placing the retrieved 3D elements around the estimated 3D cone with respect to the input sketch and the botanical structure (Section 7). A non-rigid deformation is applied to the 3D elements in a controllable manner so as to further match their silhouette to the input sketch.

4. Construction of Floral Model Database

To robustly reconstruct realistic or artistic shapes of floral objects from a sketch, our system makes use of a database consisting of different floral models either collected from the internet or from our own designs, created with a modeling software using photos as reference. Since our goal is to perform an assembly-based modeling by combining the retrieved elementary models using the sketch input, we segment each floral model into elementary botanic objects (flower centers, petals, etc.) before they can be stored in a database (see Fig. 2). Each model is associated with metadata containing the following information:

- Element type (petal, flower center, leaf, stem)
- Flower type (rose, daisy,…)
- Tip points and base points (in case of a petal)
- Attachment area (where other parts can be attached)

The labeling has been done manually, although an automatic segmentation and part recognition could be applied by using a learning based technique [KHK10]. 86 elementary models based on 20 flower models of several different types have been used in the current system.

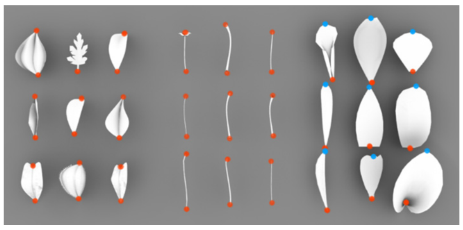


Figure 2: *Examples of elementary models in our database: Leaves(left), stems(middle) and petals(right). Attachment areas are shown in red and the petal tips in blue.*

5. Sketch-Based Retrieval of Floral Parts

The elementary floral shapes in the database are labeled on the basis of their types, along with the calculated spatial profiles. In order to retrieve a good set of elementary models by using our sketch input, we first segment the sketch curves into elementary ones, each representing a petal, a stem, a leaf, or a flower center. Our system then uses each part of the drawing as query input and retrieves 3D models from the database, which serve as elements for the reconstruction. In this section, we describe how we segment the sketch curves into elementary ones (Section 5.1) and used them as input to the sketch-based shape retrieval (Section 5.2).

5.1 Segmentation of sketch strokes

We perform the segmentation by detecting the boundary points on the sketch curve, a point of spatial or temporal discontinuity. T-junctions or cusps are natural candidates for the boundary points since they represent such discontinuity. Similarly, a start- or an end-point of each stroke represents temporal discontinuity. In most cases a floral element is drawn as a single stroke but sometimes we need to group a set of strokes into one, or inversely, split one stroke into more than one segment, as illustrated in Fig. 3(a-d). Fig. 3(a-b) show that two strokes represent a bending of a petal. On the other hand, a single stroke in Fig. 3(d) needs to be segmented into multiple curves, each representing a petal. The torsion of a petal shown in Fig. 3(c) can be drawn either in a single stroke or two. In the former case, it needs to be differentiated from the case shown in Fig. 3(d), and in the latter case, it should be merged into one segment as in Fig. 3(a-b).

In order to handle such stroke- or segment-groups, we develop the following strategies. Firstly, we check for loops in each stroke, as detected by the self-intersection points of a stroke (Fig. 3(d) and Fig. 3(c) when it is drawn as a single, red stroke). If the area inside the loop is negligible, we segment the stroke at the point of maximal curvature along the loop (Fig. 3(d)). Secondly, when two temporally subsequent strokes meet at one junction point, we consider the two following cases:

(1) If pairs of coinciding tangent directions can be found (Fig. 3(c) when it is drawn by two green strokes), we merge them as a single elementary curve.

(2) If the junction angle is more than a threshold θ, and the first stroke ends at a T-junction with itself, we assign the same petal label to the second stroke (Fig. 3(b)).

Thirdly, when two temporally subsequent strokes meet at two junction points, we check if one of the junction angles is more than a threshold, i.e. 35°, (Fig. 3(a)). If the case, we consider that they actually represent different parts of a same petal and therefore assign a same label to both.

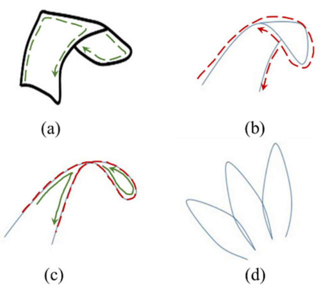


Figure 3: *Atypical cases of stroke merge or split : (a,b) Two strokes represent a bending of a petal can form a junction point or two; (c) The torsion of a petal can be drawn in a single stroke (red) or two (green); (d) A single stroke needs to be segmented into multiple curves each representing a petal.*

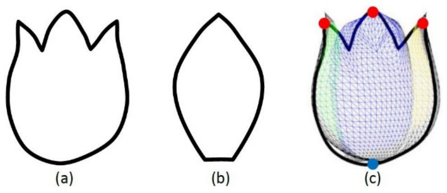


Figure 4: *Example of a sketch that cannot be segmented automatically (a). The user is asked to draw a template petal (b), which is used to retrieve the corresponding 3D shape by the system. Next, the user is requested to locate the petal tips in the sketch (red dots in (c)). The petal tips together with the base point of the cone (blue dot in (c)) are used to place the petals in the drawing (shown in wireframe with different colors).*

Template drawing. In order to perform a sketch-based retrieval of elementary shapes in the database, the sketch curve depicting the elementary object should be sufficiently detailed. When the sketch is drawn with a high level of abstraction, the aforementioned algorithm may fail to segment the sketch correctly (Fig.4(a)). This, in turn, results in a failure of the retrieval of shape elements that meet the aforementioned criteria. In such case, the user is requested to draw a template curve representing the front view of the petal (Fig. 4(b)), which will be used as a query input to the 3D database instead of the segmented petal curve from the original sketch. Since the failure of petal segmentation inherently leads to the failure of automatic petal-tip detection on the 2D petal-tip ellipse, the user instead specifies the petal tip points, by ticking directly on the sketch (Fig. 4(c)). The template petal drawing is also useful when the sketch-based retrieval fails to find a 3D petal model that meets the predefined conditions. In such cases, the user is informed about the failure and is requested to draw a template petal which will used for a new shape retrieval.

5.2 Retrieval of floral parts

The result of the segmentation, the elementary curves corresponding to floral parts, is used as input to a sketch-based shape retrieval. It is an implementation of [ERB*12], whose main idea is to compare the bag-of-features of the rendered images of a 3D model with that of the sketch. Compared to other sketch-based retrieval methods (e.g. [SXY*11]), it allows for partial matching which is necessary in our application where petals in the sketch are typically occluded by others. Subsequently, the visible part of the sketched petal alone can be used as a query input for the model retrieval. For each model in the database, a set of images is generated, corresponding to its 2D projection from 117 different viewpoints (102 uniformly sampled on the view sphere and 15 additional ones from near frontal views, as users tend to draw from a frontal- or nearly-frontal view). These images are used to generate a visual vocabulary and represent each model by a frequency histogram of visual words. This step is performed only once, and these histograms are stored in the database alongside their corresponding 3D models.

Once the sketch input has been segmented, our system represents each elementary curve as a frequency histogram of visual words. The similarity $S_{similarity}$ between the computed histogram and those present in the database can then be measured, to find 3D model candidates with high similarity scores, as in [ERB*12]. When using $S_{similarity}$ alone, however, some of these models may not correspond to expected ones, coming from different kinds of flowers or even from different botanical types. In order to efficiently prune out undesirable models, we fine-tune the search engine by assigning the following evaluation score S to each result:

$$S = S_{similarity} + \alpha \cdot S_{view} + \beta \cdot S_{model} + \gamma \cdot S_{ftype},$$

where α, β, γ are constants. For each petal curve of the sketch, we estimate its approximate orientation with respect to the view, by means of the 3D cone. This orientation should be as close as possible to the projection angle of the result model (φ, θ). To this end, we define the view score as defined by

$$S_{view} = \exp(-\|\Delta\varphi\|^2/\sigma_\varphi{}^2) + \exp(-\|\Delta\theta\|^2/\sigma_\theta{}^2),$$

where $\sigma_\varphi = \pi/8$ and $\sigma_\theta = \pi/4$ are constant.

S_{model} and S_{ftype} are respectively used to favor the redundancy of a same model and a flower type, among the results of all elementary curves. Once we obtain a number of candidate models for each curve with $\alpha = 1, \beta = 0, \gamma = 0$, we compute S_{model} and $S_{species}$ scores as measured by the number of occurrences times a constant c, where c is set experimentally to 0.2 for S_{model} and 0.3 for $S_{species}$. Then the scores of all candidates are modified and reordered by setting $\alpha = 1$, $\beta = 1, \gamma = 1$. In this way, a higher score is assigned to a candidate model element when other candidates belong to a same model or to a same flower type.

Despite this strategy, we can still obtain a set of petal models of different flower types on a same flower layer. In such a case, we choose the flower type that has contributed to the largest number of candidates.

Note that at the time of model selection we have no information on the botanical type of each elementary curve on the sketch, and the list of 3D models returned by the sketch-based retrieval could be composed of multiple botanical types (e.g. 2 petals, 1 leaf, and 7 stems). In order to determine the botanical type of each elementary curve and filter out models with different botanical types, we adopt the following step-wise method: (1) If the types of results for a 2D curve element are dominant either by 3D stems, we determine that the curve represents a stem. This can be justified by the fact that the shape of stems is clearly distinguishable from other botanical elements. (2) A curve is considered to be of petal type if it has at least one point located in close proximity to the petal-tip curve(s) and another close to the base point. (3) A curve whose results contain a 3D flower center is validated as such if it contains the base point. (4) The remaining curves are considered as leaves. After filtering out all models with different botanical types than the curve, the model with the highest score is selected as the final one (Fig. 5).

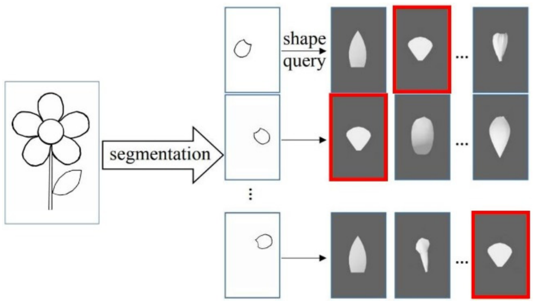


Figure 5: *For each curve representing a petal, we retrieve 3D elementary models in an order of similarity score, and select the model with the highest evaluation score combining the view consistency, frequent presence of a same flower, and of a same botanical type.*

6. Reconstruction of 3D Cone

We compute the 3D orientation of the flower by using additional information provided by the user in the form of guide strokes: (1) flower *base point* $\boldsymbol{c_2}$, the junction between the stem and the corolla, and (2) one or several *petal-tip curves* connecting the petal tips of the same layer. (Fig. 6)As shown in Fig. 6(b), the petal-tip curves may be open or closed ellipses. An open curve indicates an occlusion, either by the stem or by other petal layers (as in Fig. 6(b)). For further processing each petal-tip curve is converted to a best-fitting ellipse [FF95] or a part of it which we call as *petal-tip ellipse*.

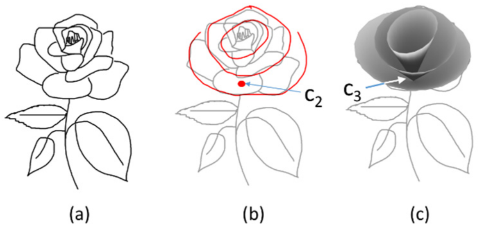


Figure 6: *After drawing the flower sketch (a), the user locates the (possibly hidden) point* $\boldsymbol{c_2}$ *where the stem joins the corolla (red dot), and draw petal-tip curves (red curves) for each petal layer (b). From these strokes, interpreted as ellipses, we reconstruct 3D cones which we will use as arrangement guides for the petals.*

6.1 Facing direction of the flower

We first determine whether the flower is oriented towards or away from the observer. Let E_2 be the largest petal-tip ellipse whose center is $\mathbf{o_2}$. We consider that the largest ellipse provides the highest precision for the global orientation computation. Let $\mathbf{s_2}$ be the end of the stem curve that is closer to $\mathbf{o_2}$. If at least one of the petal-tip ellipses is closed (no occlusion), then the flower is considered to be oriented towards the observer (Fig 7(a)).

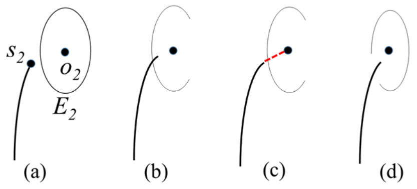


Figure 7: *The flower is oriented towards the observer if one of the petal-tip curve is closed (a) or if the stem (b) or the* [$\boldsymbol{s_2}$,$\boldsymbol{o_2}$] *segment (c) intersects with it. Otherwise the flower is oriented away from the observer (d).*

If the segment [$\mathbf{s_2}$,$\mathbf{o_2}$] or the stem itself intersects the petal-tip ellipse, then we consider that the occlusion is caused by other petal layers, therefore the flower is oriented towards the observer as in Fig.7(b) and Fig.7(c). Otherwise the occlusion is caused by the stem and therefore the flower is orientated away from the observer as in Fig.7(d).

6.2 The orientation of the corolla

We assume that all petals lie on a 3D cone whose apex $\mathbf{c_3}$ projects to $\mathbf{c_2}$ and whose base circle E_3 projects to E_2 on the sketch plane (Fig.8). We also assume that the center of E_3, $\mathbf{o_3}$ has a depth of z=0. The 3D cone axis $\boldsymbol{n_3}$ projects to $\boldsymbol{n_2}$, which corresponds to the minor axis of E_2, i.e., $\boldsymbol{n_3} = \boldsymbol{n_2} \pm d \cdot \boldsymbol{z_3}$, where d is a scalar. The second term is added if the flower faces towards the observer and subtracted otherwise (See Section 6.1). The angle α between $\boldsymbol{n_2}$ and $\boldsymbol{n_3}$ can be obtained by the ratio of r and $\|\boldsymbol{n_2}\|$, the lengths of the ellipse's major axis and minor axis, respectively. Then we have $\sin\alpha = \|\boldsymbol{n_2}\|/\text{r}$ and $\tan\alpha = d/\|\boldsymbol{n_2}\|$ which leads to $d = \|\boldsymbol{n_2}\|^2/\sqrt{r^2 - \|\boldsymbol{n_2}\|^2}$.

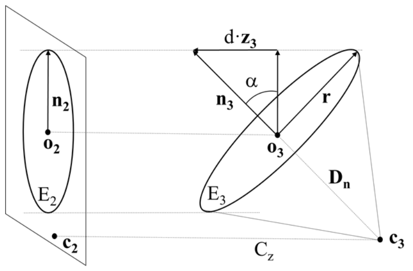


Figure 8: *A 3D disk E_3 (the base of a 3D cone) is projected onto the sketch plane as an ellipse E_2.*

Having computed the values of $\mathbf{o}_3$, $\boldsymbol{n}_3$ and $\mathbf{r}$ uniquely defining the cone's base, we now compute the location of cone's apex $\mathbf{c}_3$, which lies on line C_z passing through $\mathbf{c}_2$. It is also on D_n, the 3D axis of the flower. But the user input does not always respect the geometric constraint, i.e., these lines do not necessarily intersect in reality. Thus $\mathbf{c}_3$ is chosen as the point of $\mathbf{C}_z$ that is closest to $\mathbf{D}_n$. The cones produced in this way may have an aperture greater than 180 degrees as in Fig.1.

If the user has drawn several petal-tip ellipses, we must model several petal layers and thus several cones for each layer. We can determine the orientation of the each cone in the same way as the first cone, and translate the resulting cone so that its apex coincides with that of the first cone. In case the cones intersect, the system iteratively reorients the axis of the inner cone closer to those of the underlying cones so that the intersection can be avoided.

When the observer draws the flower strictly from the front or below, the ellipses are plain circles, where we have no depth information. For $\mathbf{r}$ and $\mathbf{n}_2$ such that $|\mathbf{r} - \mathbf{n}_2| < 0.1 \cdot \mathbf{r}$, we consider that the depth information extracted from the ellipses is not reliable and choose an aperture angle of 140 degrees for the first cone. We also assume that smaller cones are closer to the observer than larger cones. The same procedure can be applied seamlessly to cases where the flower is overly open or closed.

7. Flower Reconstruction by Part Placement

Our next goal is to construct the desired flower by placing the retrieved flower elements around the 3D cone obtained in Section 6. Petals and flower center (receptacle) are automatically positioned by using the 3D cone and their sketch (Section 7.1), as well as other parts of the flower, i.e. stem and leaves (Section 7.2). The petals are then further refined to better fit the user-drawn sketch (Section 7.3).

7.1 Placement of petals

For each 2D curve that has been labeled as petal, we first find its tip position on the 2D petal-tip ellipse by searching for the closest point on the ellipse from its petal tip. As we now have a correspondence between the 2D ellipse and the 3D cone base, we can then determine its 3D position on the cone where each petal tip must be located.

Next, we determine the orientation of each petal with respect to the cone. Assuming that the petals stem radially from the flower center, we determine two orientation angles of each petal as follows (see Fig. 9): First, the blooming angle of the petal is determined in a way that the line connecting the center of a petal to its tip is located on the cone surface. Second, the roll angle (orientation around the longitudinal axis of the petal) is determined such that the normal vector at the petal base intersects with the cone axis.

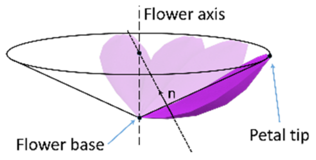


Figure 9: *Placement of petals around the 3D cone.*

Finally, if present in the sketch, the flower center is placed at the center of the petals, at the apex of the cone. Then the petals undergo a readjustment of their placements in order to accommodate well with the flower center. More specifically, we find the translation of each petal along its longitudinal direction so that the flower center and the petal base are in contact.

Occlusion handling. The aforementioned method does not consider occlusions among petals due to the facing direction of corolla. When the sketch depicts corolla with approximately half of the petals invisible due to occlusion (Fig.1), the petal-tip ellipse is left open, and an additional process is applied. We determine the approximate number of occluded petals as well as their placements on the cone as follows: We locate the petal tips of the visible petals on the 2D petal-tip ellipse and therefore on the 3D cone base, measure the average distance between neighboring petal tips on the circumference of the cone base. We then place the petal tips of the undrawn petals the average distance apart on the 3D cone base.

7.2 Positioning of the stem and leaves

We place the retrieved stem model so that its tip point is aligned with the cone apex, which lies at the center of the the corolla. The other tip point is aligned with that of the sketched stem on the sketch plane. Placement of leaves is also trivial. Assuming an orthogonal projection (i.e., 2D- and 3D-counterparts share *x*- and *y*-coordinates), we examine all the points in the attachment area of the stem and find the one whose *x*- and *y*- coordinates are closest to the 2D junction point between the stem and the leaf. Then we align the leaf base point with the 3D junction point found as above, and the tip of the leaf with that of the sketched leaf on the sketch plane.

7.3 Refinement of Petal Mesh

Each sketched petal has a shape and a bending different from each other, whereas the corolla model is obtained by the assemblage of a same 3D petal mesh. In addition, the shape of the retrieved petal model may be noticeably different from the drawing − Since our database has a limited number of flower elements, it is not always possible to find the flower elements that fits perfectly to the sketch input. In order to generate a flower model that is faithful to the 2D drawing, we deform each petal as follows.

Point correspondence between 2D drawing and 3D petal. To perform the non-rigid petal deformation, we first compute a target location for each silhouette point on the 3D petal model. This is achieved by establishing a point correspondence between the petal stroke and the silhouette of the petal model.

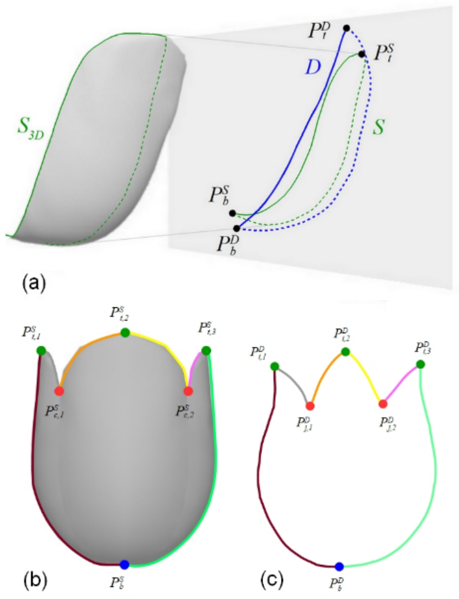


Figure 10 *: In (a), S_{3D} is the 3D silhouette of the petal and S is its projection onto the sketching plane. D is the hand-drawn curve of the petal. P_t^S, P_b^S, P_t^D and P_b^D are feature points of S and D respectively. The two segments shown in the dashed and the solid lines are sampled such that they have the same number of vertices. In (b) and (c), a similar process is applied for the case of a single stroke representing the corolla.*

Let D denote the outer contour of a given petal (Fig. 10(a)), and let P_t^D and P_b^D be the tip point, and the base point of the petal, respectively. These points can be calculated by using the petal-tip ellipse and its center point provided by the user. Let also S denote the projection of the silhouette curve S_{3D} of the corresponding 3D petal model onto the sketching plane. The feature points on S, P_t^S and P_b^S, can be found by the projection of the petal tip and the petal-base points of the petal model, which can be retrieved from its associated metadata in the database. We compute the point correspondence between S and D by first matching their tips (P_t^D to P_t^S) and base points (P_b^D to P_b^S). Then each curve segment is uniformly sampled along the arc length so that each segment contains the same number of points and for each sampled point on S its correspondence on D is found in an order preserving manner.

In case where a single curve represents several petals, we apply a different strategy. Let D be the sketch curve corresponding to the silhouette of the corolla where the automatic segmentation has failed (Fig. 10 (c)). We identify feature points on D, by combining the user-provided petal tips $P_{t,i}^D$ (i=1,...n), flower base P_b^D, and automatically detect corner points $P_{c,j}^D$ (j=1,...n-1) [HS88]. We also calculate curve S (Fig. 10(b)) which is the outer curve of the projected silhouette of the 3D flower. The location of the petal-tip points $P_{t,i}^S$, (i=1,...n), and the flower base P_b^S are obtained from the metadata of the 3D petal model, and the corner points $P_{c,j}^S$ (j=1,...n-1) are found by computing the intersection point of the two curves originating from neighboring petals. After computing the correspondence between the two sets of feature points on D and S via order preserving assignment [SN06], we generate dense correspondences by regular arc length sampling on each segment.

Laplacian deformation. Now that we have computed the point-to-point correspondence between the silhouette of the petal mesh and the corresponding stroke, we deform the mesh so as to move its silhouette points to their corresponding target locations on the input stroke. Using a class-specific deformation such as [PFZG10] would require considerable effort in pre-processing, as a deformable class for each petal type should be pre-computed. Thus, we have chosen to use a generic, Laplacian deformation [SCL*04] for its ability to preserve the details of the mesh through the deformation, in addition to its inexpensive, linear computation. The coordinates (x,y) of a point on the silhouette curve S_{3D} are aligned with the coordinates of its corresponding counterpart on curve D, while its z-coordinate remains unchanged. The specifications of the silhouette points serve as constraints of the Laplacian deformation of the whole petal model (see Fig. 11). Note that in case of an unclosed petal sketch due to occlusion, only the feature points of S along with their correspondences in D are used as constraints. Due to the smoothness energy of the Laplacian deformation, the unmatched parts of the model will remain as close as possible to the initial shape.

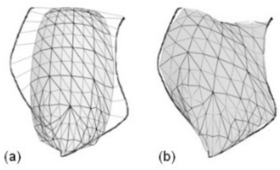


Figure 11: *For each silhouette point on the petal mesh a corresponding point is found on the sketch curve* (a)*. The refined model after the deformation* (b).

8. Results

Our modeler has been implemented in C++ as a stand-alone application running on the Windows platform. Experiments were performed on a workstation with 16GB of memory and Intel Core i7-6700 processor running at 3.40 GHz. Fig. 12 shows the user interface of our system. The flower as well as the guide strokes is drawn in the left window, using a digital pen. The user can also import an existing vector-graphics drawing into that window. The right window is a 3D viewer for visualizing the reconstructed flower mesh.

Our system has been evaluated using a variety of sketches representing different types of flowers from different viewpoints. Users from various age group 10 to 46 have participated, all of whom with little or average drawing skill. Fig. 13 shows the results we obtained using our modeler. Flowers having both single layer and multi-layers ('withered', 'imaginary', and 'lion hair') have been successfully reconstructed. The system also works well with flowers sketched from different viewing directions ('nine petals' drawn from the back view, 'tulip' and 'withered' from the side) and with various degrees of booming (from close-to-closed 'tulip' to overly open 'withered'). Remarkably, our system is able to handle more sophisticated flowers, as well as flowers whose petals fold on themselves (See the results shown in Appendix A). Note that the naive sketch of 'tulip' is well handled by our system, with the additionally drawn template petal. However, since the system automatically adds the fourth petal model that has not been drawn in the original sketch, the projection of the reconstructed model leads to a slightly different silhouette from the 2D sketch.

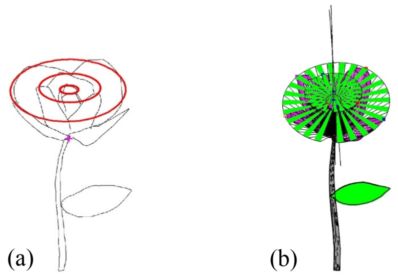


Figure 12: *The user interface is composed of two windows. The first window shows the user sketch with the flower center (pink cross) and the three ellipses computed from the petal-tip curves provided by the user (a). The second window shows the reconstructed model in wireframe and the 3D base of each cone (circle partially filled in green) (b).*

In some cases the petal model selected by the sketch-based retrieval seems quite different from the sketch (See the 2nd column of 'lion hair' in Fig. 13). This is due to the presence of a large number of occluded petals. As explained in Section 5.2 our sketch-based retrieval module searches for a model for each petal, even for occluded ones, thanks to its partial matching capability. Therefore, when the sketch contains many petals showing only their tips, a petal model that best matches those tips will be chosen, even if it has a slightly different shape from the non-occluded petals in the sketch. However, the subsequent deformation step corrects well this shape difference, as can be observed in Fig.13.

The time required for the sketching depends on the level of drawing skill and the complexity of the flower and may take from a few seconds to five minutes. Once the full sketch had been provided, only a few seconds of interaction was necessary, even for our most complex model ('lion hair' in Fig. 13). All subsequent computations, i.e. segmentation of the input sketch, retrieval of elementary shapes from the database, and their placement on the estimated 3D cone took a few seconds (20 seconds for the 'lion hair' which contains 44 parts). Note that the flowers requiring the largest amount of interaction time are those on which the stroke segmentation fails ('tulip' in Fig.13), as the user has to draw a petal template and provide a few more clicks on the petal tips.

8.1 Limitations

One limitation pertaining to our system is the expressibility of the reconstructed geometry that depends on the variability of the model database. Although the deformation of petals greatly reduces this problem, there can still be the case where the sketched petals are so different from any petal mesh in the database that the deformation step fails to find acceptable correspondences to produce expected results.

Another limitation of our technique, inherent from the single-view setting, is revealed when convex petals are sketched from a strictly frontal view. In such case, the segmented curves of the drawing do not represent well the 3D curvature of the petals, and the sketch-based retrieval will return petal models whose 2D projection will maximize the similarity with the sketch. These petals can be rather flat, and may not necessarily correspond to what the user intended. Fortunately, users tend to choose more 'informative' views in such case, i.e. non-frontal views naturally revealing the curvy petals. We also note that the presence of the flower stem in the sketch is obligatory, since we make use of the relative position between the corolla and the stem when determining the orientation of the corolla.

9. Conclusion and Future Work

In this paper, we have presented a modeler for floral objects that takes 2D sketch as input. Our system has shown to perform robustly on a broad set of sketches drawn by several users without special training, successfully reconstructing flowers having single or multiple layers, from different viewing directions and with various degrees of blooming, as well as more sophisticated flowers. We make use of the fact that people tend to simplify the structural and geometric complexity of floral objects in their 2D drawings. We also profit from the fact that the floral objects share a common structure which is a relatively simple geometric form (i.e. cone), whose parameters are computed from the user-supplied guide strokes. Reliable shape results are achieved by making use of a shape database and sketch-based shape retrieval. To the best of our knowledge, this is the first presentation of a sketch-based modeler for flowers that is able to produce quality results from a single view in a robust manner, with minimal user interaction.

Our modeling technique offers a number of advantages over existing works. Firstly, modeling flowers with our system is extremely rapid and is as simple as drawing the flower silhouette on the screen, followed by a few additional guide strokes. Our system is able to produce results more efficiently compared to other interactive methods (e.g. Ijiri et al. [IOOI05] requires 40 mins for a sunflower), and with similar time cost as Yan et al. [YGC*14] with the freedom of view selection (unlike [YGC*14] where flowers photos are taken from the front or slightly oblique view). Secondly, it is a truly sketch-based tool. Since the drawing and the modeling is entirely made in a single view without any interactive manipulation or the placement of 3D parts on the flower, (unlike Ijiri et al. [IOOI05]) our tool keeps the simplicity of drawing on the paper. This increases the usability of our system by nearly everyone. Finally, it enables the creation of a wide variety of flowers including those with multiple layers of petals, and can handle the sketch of highly abstracted drawing style ('tulip' in Fig.13).

In the future we plan to further improve the quality of the result by extending the deformation to non-corolla parts of the flower. Another idea would be to use the colors present in the sketch in order to automatically generate and apply textures to our model. It would also be interesting to develop a learning-based method that can reconstruct flower models from sketch, overcoming the limitations (Sketch is restricted to the same view used in the training data.) of the current state of the art techniques, e.g. [LGK*17, HKYM17].

Acknowledgements

This research has been supported by the French Minister of Higher Education, Research, and Innovation.

References

[ACP03] Allen B., Curless B., Popovic Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans. Graph. 22,3 (2003), 587-594.

[BV99] Blanz V., Vetter T.: A Morphable Model for the Synthesis of 3D Faces. Proc. ACM SIGGRAPH 1999.

[DC10] De Craene, L.P.R. 2010. Floral Diagrams: An Aid to Understanding Flower Morphology and Evolution. Cambridge University Press.

[ERB*12] Eitz, M., Richter, R., Boubekeur, T., Hildebrand, K., and Alexa, M., Sketch-based shape retrieval. ACM Trans. Graph. 31, 4 (2012), 1-10.

[FF95] Andrew W. Fitzgibbon, Robert B. Fisher: A Buyer's Guide to Conic Fitting. BMVC (1995), 1-10.

[FKS*04] Funkhouser T. A., Kazhdan M. M. , Shilane P., Min P., Kiefer W., Tal A., Rusinkiewicz S., Dobkin D. P.: Modeling by example. ACM Trans. Graph. 23, 3 (2004), 652-663.

[GLX*16] Guo X., Lin J., Xu K., Chaudhuri S., Jin X.: CustomCut: On-demand Extraction of Customized 3D Parts with 2D Sketches, Comput. Graph. Forum, 35, 5, (2016), 89-100.

[HKYM17] Huang H., Kalogerakis E., Yumer E., Mech R.: Shape Synthesis from Sketches via Procedural Models and Convolutional Networks. IEEE Trans. Visualization and Computer Graphics 2017.

[HS88] Harris, C., and M. Stephens: A Combined Corner and Edge Detector, Proc. 4th Alvey Vision Conference (1988), 147-151.

[IOI06] Ijiri T., Owada S., Igarashi T.: Seamless Integration of Initial Sketching and Subsequent Detail Editing in Flower Modeling. Comput. Graph. Forum 25, 3 (2006).

[IOOI05] Ijiri T., Owada S., Okabe M., Igarashi T.: Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. ACM SIGGRAPH 2005.

[IYYI14] Ijiri T., Yoshizawa S., Yokota H., Igarashi T.: Flower modeling via X-ray computed tomography. ACM Trans. Graph. 33(4): 48:1-48:10 (2014).

[KHK10] Kalogerakis E., Hertzmann A., Singh K.: Learning 3D mesh segmentation and labeling. ACM Trans. Graph. 29, 4 (2010), 102:1-102:12.

[LF08] Lee J., Funkhouse T. : Sketch-Based Search and Composition of 3D Models. In Proc. Eurographics conference on Sketch-Based Interfaces and Modeling (2008), 97-104.

[LGK*17] Lun Z., Gadelha M., Kalogerakis E., Maji S., Wang R.: 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. Proc. International Conf. 3D Vision (3DV) 2017.

[PFZG10] Prasad M., Fitzgibbon A.W., Zisserma, A., & Gool L.V. Finding nemo: Deformable object class modelling using curve matching. IEEE Conf. Computer Vision and Pattern Recognition, 2010.

[QTZ*06] Quan L., Tan P., Zeng G., Yuan L., Wang J., S. B. Kang S. B.: Image-based plant modeling. ACM Trans. Graph. 25, 3 (2006), 599-604.

[SAG*13] Shtof A., Agathos A., Gingold Y. I., Shamir A., Cohen-Or D.: Geosemantic Snapping for Sketch-Based Modeling. Comput. Graph. Forum 32, 2 (2013).

[SCL*04] Sorkine O., Cohen-Or D., Lipman Y., Alexa M., Rössl C., Seidel H.-S. Laplacian surface editing. In Proc. Eurographics/ACM SIGGRAPH Symp. on Geometry Processing (2004), 175-184.

[SI07] Shin H., Igarashi T.: Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. Proc. Graphics Interface (2007), 63-70.

[SM04] Seo H., Magnenat-Thalmann N.: An example-based approach to human body manipulation. Graphical Models 66, 1, (2004), 1-23.

[SN06] SCOTT C.,NOWAK R. D.:Robust contour matching via the order preserving assignment problem. IEEE Trans. Image Processing 15, 7 (2006).

[SXY*11] Shao T., Xu W., Yin K.,Wang J.,Zhou K., Guo B. Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching. Comput. Graph. Forum, 30 (2011).

[XCF*13] Xu K., Chen K., Fu H., Sun W.-L., Hu S.-M.: Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. ACM Trans. Graph. 32, 4 (2013), 123:1-123:15.

[XXM*13] Xie X., Xu K., Mitra N. J., Cohen-Or D., Gong W., Su Q., Chen B.: Sketch-to-Design: Context-Based Part Assembly. Comput. Graph. Forum 32, 8, (2013).

[YHG*16] Yin K., Huang H., Long P., Gaissinski A., Gong M., Sharf A.: Full 3D Plant Reconstruction via Intrusive Acquisition. Comput. Graph. Forum 35, 1, (2016).

[YGC*14] Yan F., Gong M., Cohen-Or D., Deussen O., Chen B.: Flower reconstruction from a single photo. Comput. Graph. Forum 33, 2 (2014), 439-447.

[ZYFY14] Zhang C., Ye M., Fu B., Yang R.: Data-Driven Flower Petal Modeling with Botany Priors. Proc. IEEE Computer Vision and Pattern Recognition (2014).

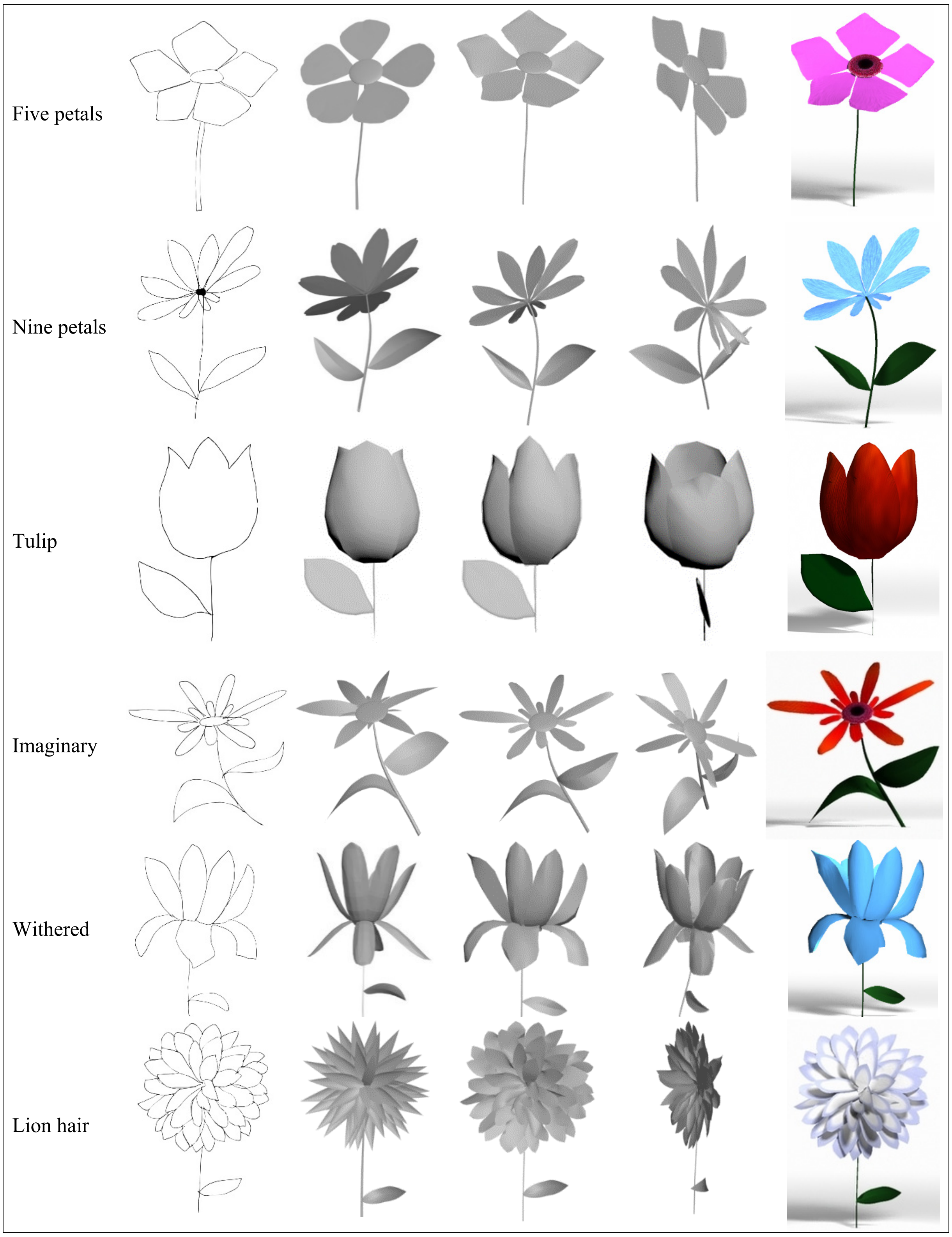
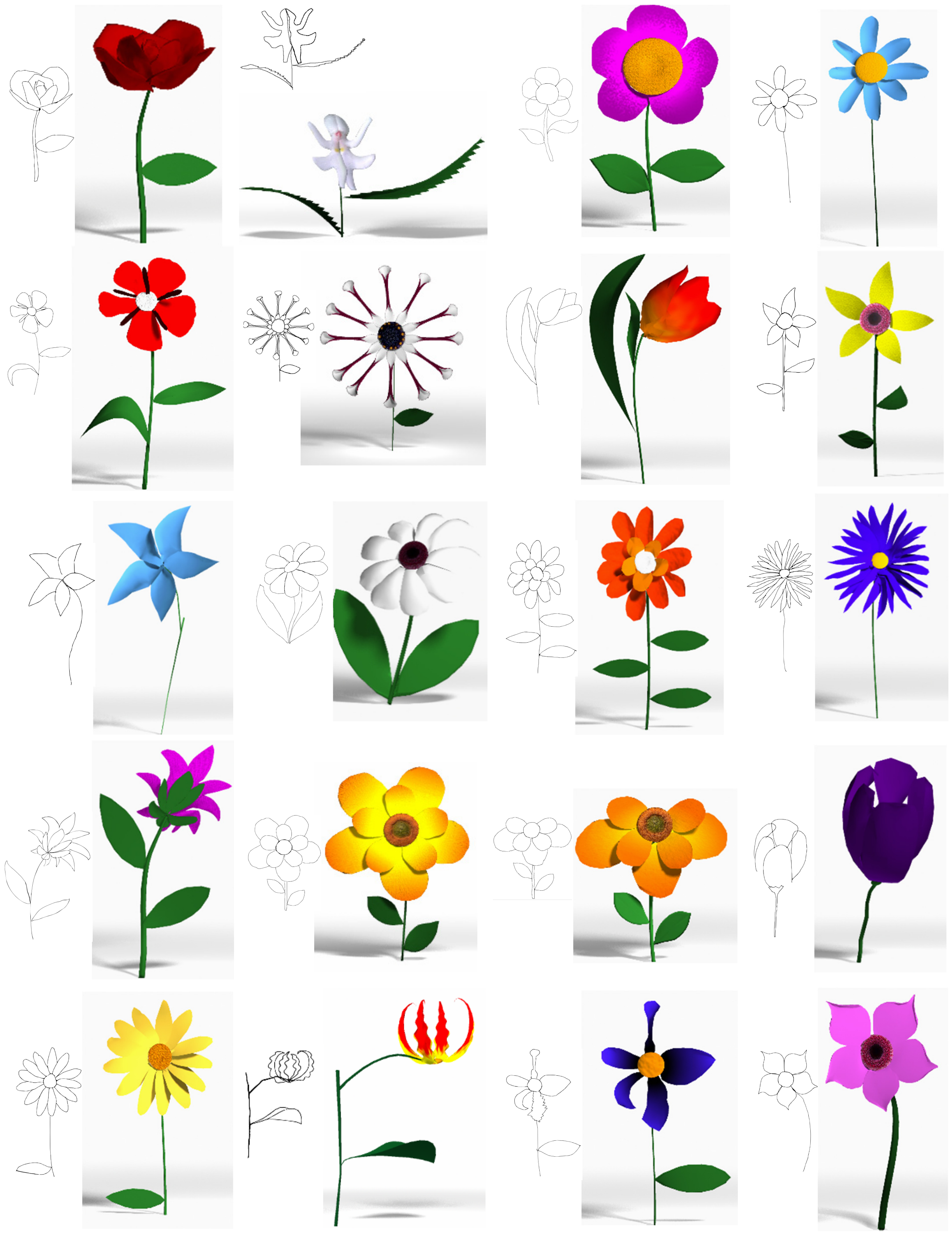


Figure 13: *Results obtained with our modeler. From left to right: the input sketch; the reconstructed models by assembly only and with the deformation from the same view as input; the models viewed from a different direction and with texture mapping.*

Appendix A: More flower model examples from our modeler



Appendix B: A subset of the elementary flower models in our current database