EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2010)
M. Alexa and E. Do (Editors)

Modeling 2.5D Plants from Ink Paintings

Cristina Amati and Gabriel J. Brostow

University College London, UK
http://visual.cs.ucl.ac.uk/pubs/sumi-e

Abstract

Chinese ink painting, known as sumi-e, is a traditional art form based on careful placement of expressive brush strokes. To extend such brush strokes from their paper form, we propose a system that can digitize and transform such paintings into 2.5D sprites, while preserving the artist's unique style. We apply our paintstroke-extraction technique to plants, the most traditional subject of sumi-e, and one where CG modeling is fairly hard, but procedural or simulated animation techniques can subsequently be brought to bear. Instead of forcing artists to use pressure-sensitive digital tablets, we address the problem of how to non-invasively decompose a real painting of a plant into semantically meaningful components, like leaves and petals. Webcam filming of the artist at work provides valuable cues. This part of our system needs no human interaction and no prior training, and achieves its goal solely by analyzing the video timeline to find even intersecting brush strokes. Afterwards, it constructs a model of the plant and textures it with the extracted brush strokes, inpainting when necessary.

1. Introduction

We want to create useful digital models directly from hand painted art, using a system that is non-invasive and largely automatic. Because the breadth of traditional art techniques would make our task intractable, we have chosen to focus on a particular art-form and a restricted painting subject. Sumi-e is a very delicate and expressive form of art developed by Chinese monks in the seventh century. Traditional sumi-e themes are mainly nature themes. We have focused our attention on paintings of plants due to their physical structure.

Many drawing and photomanipulation platforms offer brushes that mimic the sumi-e style. Adobe Photoshop includes a sumi-e filter and brushes that create a wide, wet brush stroke "with a Far Eastern flavor" [BF03]. Nevertheless, the authenticity and complexity of the hand painted art could currently not be recreated digitally. An artist uses *big brushes* and a variation of pressure and ink concentration which can not be reproduced by using graphics input tablets. [KKK03] present an outstanding system that renders sumi-e representations of 3D models. Also, many approaches exist for simulating calligraphic brush strokes [SXSC02] or other painterly styles [GCS02].

Our system can help us benefit from the experience of the masters directly, by detecting and extracting their brush strokes, and then reconstructing them in a 2.5D environment. This is done using a video recording of the artist at work, and it requires no additional training or manual annotations. Through several innovations, we empower sumi-e artists to create models and animations with almost no added effort or computer skills.

(a) bamboo (b) iris (c) orchid

Figure 1: Examples of traditional sumi-e plant paintings (Copyright owned by the authors).

Our prototype expands on known segmentation techniques to extract components of the artwork's ink. This is hard because of user-occlusions and overlap within the artwork due to the unpredictability of the artist's movements. Video analysis of the artwork's progression reveals the order in which components were painted, and a single high-resolution digital scan of the finished painting captures the brush stroke details. Further on, leveraging the planar nature

DOI: 10.2312/SBM/SBM10/041-048

of most plant components, we construct a billboard model and texture it with the extracted brush strokes. Obtaining the footage of the artist painting does not require a special studio setup. A simple webcam with a resolution of 640×480 is enough to yield good quality results. One of the major requirements of our project is to not hinder the creative process of the artist in any way. To our knowledge, this is the first attempt to convert actual ink brush strokes into digital form, though certainly previous efforts have explored fitting of brush stroke priors to paintings [XXK*06].

Our main contributions are sub-algorithms to i) Segment a painting using the recorded video of its creation, ii) Extract key elements and their hierarchy from the video frames, without imposing constraints on the artist such as asking them to take their hand out of the scene after painting each element or limiting the drawing pace, and iii) Reconstruct the painted plants using a plugin in a 3D environment, using a shape constraint model which mimics real plants and can be applied to most sumi-e traditional plant themes.

2. Related Work

A very eloquent work on synthetically creating brush strokes similar to the ones used in calligraphic lettering and paintings is presented by [SXSC02]. Their solution models a brush stroke as an interval spline with constraints between knots. They produce a vectorized output that is resistant to scaling. Also, their model is able to capture variation in thickness across a single stroke. Brush stroke modeling has also been implemented by [HL94]. We concentrate on extracting the features from the painting itself without relying on synthetic methods.

The research we have found to be most similar to our own is that of [XXK*06]. Their major contribution is the automatic recovery of vectorized brush strokes from the final painting. To achieve this, they segment the image and then fit primitive brush strokes on the individual segments using a stroke database created by an experienced sumi-e painter. As an application of their system, they create an interface for brush stroke based animation of the paintings with spectacular results. Just like in our case, they present a solution for separating overlapping strokes. Although our goals are similar, the major difference is in the execution. We require no additional database, and non-invasively "observe" the artist to capture the structure of their painting. Further, our segmentation take a matter of seconds and is done automatically based on processing the video of the artist at work, instead of requiring user interaction.

Other efforts in the area of creating animations from a single image that are not directly related to the sumi-e style have been made by Chuang et al. in 2005. Their approach aims at animating arbitrary scenes that respond to natural forces by using stochastic motion-textures. The motion-texture is a time varying 2D displacement map [CGZ*05]. In their approach segmentation is user assisted because they take into consideration arbitrary scenes. The current project tries to bring an automated segmentation of the pictures by using the recording of the drawing process. This is possible because we have chosen to narrow down the drawing style and subject range.

Another solution for animating still pictures is proposed by [HDK07]. Unlike Chuang et al.'s work, Hornung et al. concentrate on animating human-like characters from pictures by fitting them with a 3D skeleton and 3D motion data. Our work differs in the approached subject, that of plants and the fact that we do not seek to fit any data to our models but rather define a set of constraints to mimic plant movement. The similarity consists in the concept of upgrading from two dimensionality in order to give a better representation of an object.

Fei et al.'s work [FLJX08] aims at combining the aesthetics and intuitiveness of 2D drawing and sketching with the six-degree freedom provided by 3D animation. Fei et al. advanced the observation that 3D animation and storyboarding should benefit from the user's strong 2D drawing skills.

Modeling trees and plants from reference photographs with limited user interaction has been successfully attempted by [NFD07], [TZW*07], [QTZ*06] and [RLP07]. Although we admire the complexity of the models that have been built only from images, our goal is slightly different in that we want to render non-photorealistic plants and we only have one image that is a stylized representation by an artist, therefore constructing a full 3D model would be an underconstrained problem. The work of [OOI06] addresses the problem of constructing 3D tree models from user input sketches by inferring depth.

One inspiring research that is not directly related to our own but makes use of cameras and projectors together with traditional art is that of [FR06]: *Projector-Guided Painting*. They use projections on canvas and motion tracking to help novices create a painting step by step and we make use of cameras for the inverse problem: breaking the painting into elements using the creation steps. [BJS*08] also use video input from the user for animating paper cutouts. Although very impressive, this paper addresses character animation which is very different from our subject.

3. Our Algorithm for Video Based Modeling

Our algorithm works in a series of steps to combine the high resolution spatial content of a scanned painting with the temporally informative cues in video of the artist at work. To reach the goal of a *stylistically accurate* 2.5D plant model, our system can be outlined in the following steps:

1. Find correspondences and register the scan against frames of the video,
2. Detect and segment the ink's mask in a sliding temporal window of video frames,

3. Locate *keyframes*, video frames where new paint has been added,
4. Extract individual plant components from the keyframes and *correct* them,
5. Apply the low-res component models to the high-resolution scan.

Each step is detailed in this section, and together, they enable us to make a 2.5D model of the painted plant.

3.1. Registration

Offline registration can be applied repeatedly to the video to align all frames, in case the canvas or camera is repeatedly bumped or moved. The arbitrary position of the camera relative to the drawing surface can be modeled as a homography because the canvas is planar. This is ideal for our case, because we can consider at least the four corners of the paper as the correspondence points, and any painted features (matched using SIFT features [Low04] for example) would simply help to overconstrain the solution.

To find the projective transform even of the blank canvas, the system must localize the paper's corners in a video frame. This process is carried out in using the Hough transform to find the paper boundaries (dominant edges), and intersection of these lines to find sub-pixel corner locations. Having the paper corner coordinates for a video frame, we can now compute the projective transformation between them and the scanned image corners [HZ00].

3.2. Frame Processing

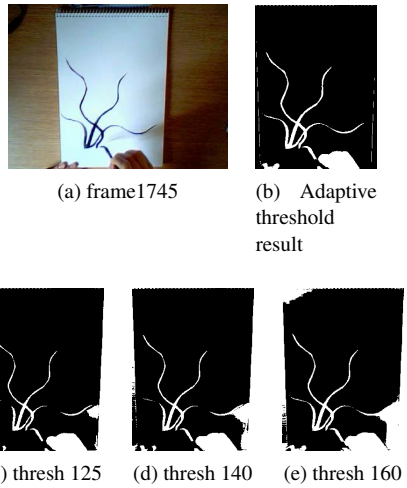
Because we are working with monochromatic ink paintings on white paper and the contrast that this implies, we assumed that a first step of thresholding would help identify the ink. A fixed threshold is not appropriate due to the changes in lighting conditions during the painting process. We use an adaptive threshold that is calculated for each pixel based on its neighborhood, so

$$dst(x,y) = \begin{cases} maxValue & \text{if src(x,y)} > \text{T(x,y)} \\ 0 & \text{otherwise,} \end{cases}$$

where $T(x,y)$ is computed as a Gaussian weighted sum over a $blockSize \times blockSize$ pixel neighborhood. We found $0.5 \times min(frame_{width}, frame_{height})$ to be a good block size. The block needs to be big enough to filter out noise and shadows cast on the paper.

3.2.1. Detecting the Ink

Simply thresholding does a fairly good job of separating the paper from the other objects in the frame, however it does not pick out only the ink from the image. Figure 2 shows examples of a thresholded frame containing a part of a painted plant, but also a hand, the brush, two fingers and some additional shadows, that we refer to as *other objects*.



(a) frame1745 (b) Adaptive threshold result

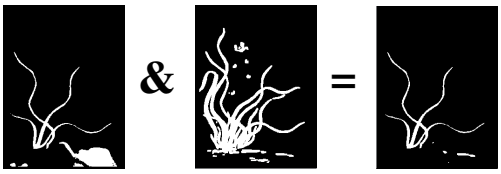
(c) thresh 125 (d) thresh 140 (e) thresh 160

Figure 2: **Adaptive vs. global thresholding.** The first row shows an original video frame and the adaptive threshold result we use. The second row shows the result of trying different global thresholds: 125, 140, 160. The low thresholds miss out important details, like tips of leaves, while the higher thresholds incorporate shadows into the set of painted elements.

Our non-invasive artist observation method allows for low resolution filming from an arbitrary camera position, uncontrolled lighting conditions, and free movement of the artist. Since the system uses no prior training data, it is a formidable task to recognize the other objects in the scene. We have experimented with many methods for telling objects apart, including color analysis and skin segmentation, but these have proven to be unstable because of the arbitrary lighting. We did find that temporal *stability* was a persistent characteristic of the ink.

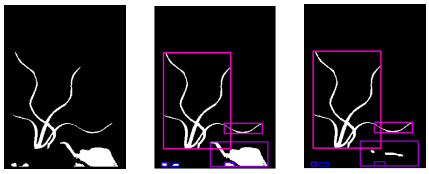
The final algorithm that delivers good detection of ink outlines follows, where each video frame already includes its homography-based registration to the last frame:

1. *Input:* current frame and last frame thresholded, cropped and cleaned of any paper margin artifacts
2. **Take logical AND between current frame and last frame**
   Painted objects are stable across subsequent frames while hands, brushes, and shadows are constantly moving (Figure 3).
3. **Detect contours on current frame**
4. **Compute similarity measure using Hu Moments**
5. **Accumulate matching contours**
   There may be more than one contour that we need to preserve. Particularity in sumi-e, a bent or turning leaf can be represented as two discontinuous segments. Accumulating contours is done based on contour matching of the two subimages described earlier, using the first 7 Hu mo-



(a) Current frame (b) Last frame (c) Logical & result

Figure 3: Because there can be more objects (hands, brush, strong shadows, etc.) in the currently processed video frame than just the painted object, we take a logical-& between the current frame and the last frame, to detect which frame components are stable.



(a) (b) (c)

Figure 4: Contour detection. From left to right: the current frame, detected contours on the current frame shown in colored bounding boxes, and contours detected on the AND operation with the last frame. The purpose is to identify separate objects in the current frame, to establish which of them is stable.

ments. Given the contours C_1 and C_2, the similarity measure is computed with the formula:

$$S(C_1,C_2) = \sum_{i=1}^{7} \frac{|m_i^{C_1} - m_i^{C_2}|}{|m_i^{C_1}|} \quad (1)$$

where $m_i^{C_k} = sign(h_i^{C_k}) \cdot log(h_i^{C_k}), k \in 1,2$ and $h_i^{C_k}$ is the i^{th} Hu moment of the contour k. We then threshold the value of S with $T = 0.1$, $S = 0$ being perfect similarity. Figure 5 shows the chosen contours and the final output of this algorithm.

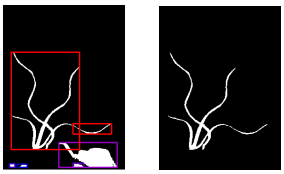


Figure 5: The contours belonging to the painted plant shown with red bounding boxes are accumulated for the final output of the ink detection algorithm. The left image shows the automatically chosen contours and the one on the right shows the algorithm output.

6. *Output:* an element containing the detected ink, if any.

3.3. Finding Keyframes

The task in this part of the algorithm is to identify *keyframes*. A keyframe contains a newly completed component of the painted plant without the hand, brush, or any other object. As in Figure 6, it is possible for non-ink to be in the keyframe, provided it is not directly connected to the element we want to segment.

Usually one or two brush strokes are sufficient to create one part of a plant such as a leaf, petal, branch, etc. This is why the logical components of the object derive naturally from the painting process which we record. This enables us to look through the timeline of the video and retrieve the frames containing those completed objects.

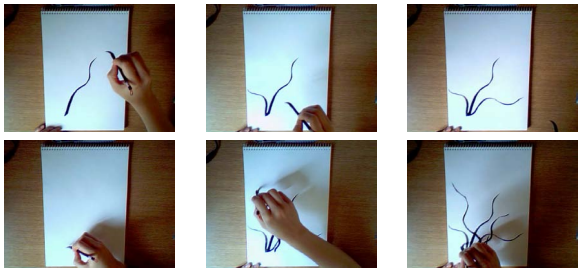
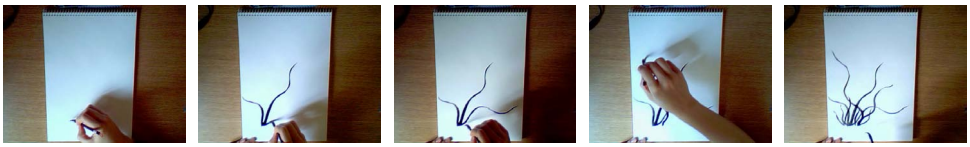


Figure 6: First row: 3 automatically detected *keyframes*- unoccluded frames that contain a new finalized element of the painting (in this example a new leaf); Second row: non-keyframes

We devised an iterative process that consists of a number of filtering stages. This helps gradually narrow down the possible candidates. The input to this part of the algorithm is the stack of frames processed with the method described Section 3.2, images that contain only a white mask of the current element on a black background.

The stages of iterative eliminating process are:

1. **Suppress empty masks**
   As in Section 3.2 the masks for some frames are just black images. This occurs in the beginning of the painting process where for many frames there is nothing on the canvas (Figure 7a and 7f) or when the hand or the brush is directly connected to the painted elements, making it thus difficult to recognize the contour.
2. **Suppress masks that have a lower pixel count than the previous ones**
   It is logical that as we advance in the video timeline we should have more pixels added to the drawing. This is reflected in the extracted masks as well. The fact that one mask has fewer pixels than the previous one tells us that it was partially occluded, therefore it should be suppressed. In Figure 7, 7c is suppressed because its mask 7h has a lower pixel count than the previous mask 7g. This happened because the hand with the brush in 7c is connected to the painted object and therefore the contour was not properly detected.
3. **Detect progression and suppress intermediate masks**
   When the progress of painting one image element is visible, the system so far considers each intermediate state as a separate keyframe. This leads to unwanted results



(a) frame 90 (b) frame 905 (c) frame 1330 (d) frame 1660 (e) frame 2985

(f) mask 90 (g) mask 905 (h) mask 1330 (i) mask 1660 (j) mask 2985

Figure 7: First row: original video frames; Second row: corresponding masks of the ink outline. It can be seen that many frames have missing or incomplete ink outlines. Only 7e can be considered a keyframe because its mask 7j corresponds exactly to the ink outline.

in the final stage. Using a proximity measure, we detect this case and suppress all but the final completed element mask.

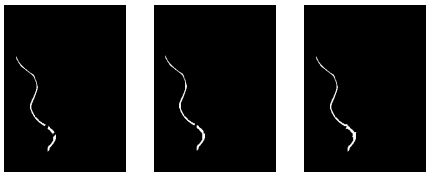
4. **Suppress similar masks**
   If after the previous steps there still are frames masks that are highly similar, we shall choose only one of them as the keyframe. This situation might occur when the hand is only hovering over the painting without drawing anything, and then uncovers the same painted elements as before in a later frame.

3.4. Element Extraction from Keyframes

In this step, we focus on extracting the individual painted plant elements or components from the keyframes. Valid components would be: leaves, petals, stems, etc. Sometimes, turning leaves can be painted using two brush strokes. Knowing the way sumi-e is made, these two brush strokes are completed subsequently, giving us a hint in the timeline that they should be treated as one.

Some elements naturally occlude others. The first element to be painted can be retrieved in its entirety. Retrieving each further component is based on image-differencing. Before correction, one or more gaps appear in the newer processed elements.

Erosion and dilation are insufficient to compensate for gaps. The inpainting function provided by OpenCV [BK08] accounts for minor gaps and requires a mask to localize it in the image. If the mask does not exactly match the gap, the inpainting fails. The OpenCV inpainting algorithm uses the Telea method. It is not perfect, but just satisfactory for this stage. Artifacts are somewhat hidden because overlapping brush strokes occlude each other in the reconstruction.



(a) Element after frame subtraction (b) Element after dilation and erosion (c) Element after inpainting

Figure 8: The first image shows the element having two gaps. Repeated dilations and erosions can only fill smaller gaps and not completely. Inpainting using the previous elements that have caused the gap to appear can unite the elements across larger gaps but with some of artifacts.

3.5. Element Extraction from the Painting

The final extraction of the plant components from the original image is based on the masks retrieved from the video frames. The reason for this is the fact that the colors of all elements are almost the same, which would make any color based segmentation go wrong. On the other hand, our goal is to extract semantically meaningful elements from a completed painting. With no prior training, semantically based segmentation systems could also fail at this stage, because adjacent regions are too similar with respect to the same characteristics to tell apart. Probably, at this stage, only a machine learning approach could make such a difference between image elements. This would however require advanced algorithms and training which is not the subject of this research. Furthermore, in a very cluttered image it is almost impossible to tell individual elements apart without having a notion of the timeline of their creation.

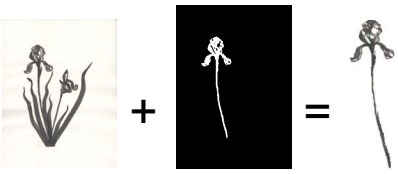
The method we used is fairly simple but it relies heavily on the correctness of the projective transformation. The masks from the video frames are inflated to the high resolu-

tion of the scanned painting by using the projective transform together with an interpolation method. Finally, the mask is applied on top of the original image and the element is extracted onto a new image with white background.

The problems that might arise here are numerous, however, with a good registration, the method is quite robust on all the tested images. The key is to get the image perfectly aligned with the mask. This, though is also a problematic task, firstly because of the great difference in scale, then because of the numerous operations applied to the images. Thresholding might cause minor information loss and the interpolation stage could also bring slight modifications to the final resulting mask. Any slight shifts could alter the extracted element.

Another encountered problem was one specific to ink and wash paintings: transparency. Every new stroke adds more concentration to the color on the paper, which makes overlapping or over-painted regions more obvious. However, this is the charm of this style and it will be preserved as it is in the final result.

During this stage, a configuration script is written.



(a) Scanned painting (b) Mask from video (c) Result

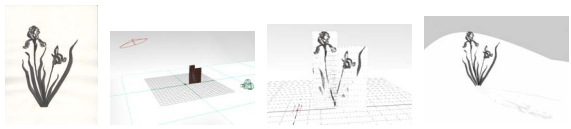
Figure 9: Extracting each element from the final painting using the masks retrieved from the video analysis.

We now have a stack of high resolution painting elements and a configuration file that tells their offset in the original image. Having come this far, one can see in these final images the actual brush stroke intact with all its nuances and texture that makes hand drawn art so unique.

3.6. Model Construction

For constructing a plant model from our segmented painting, we have to make a weighting of the possibilities with our goals of plausibility and automation. We have chosen the Maya Framework as a modeling environment. Amongst things that we have tried is using textured billboards, transforming the exported stack of painting elements into meshes of triangles using Maya's texture to geometry function and finally an improvement to the textured billboard version.

Representing plants and trees with billboards is not itself novel. Because of the high frame rate constraints, trees in games are often rendered as image impostors. Billboards



(a) Scanned painting (b) Reconstruction in Maya (c) Rendered camera view (d) Rendered scene

Figure 10: This figure shows the reconstruction with cropped billboards in Maya. The scene is built use one single command that takes as argument the location of the folder containing the segmented images and the configuration file. The last image shows the rendered result of the reconstruction.

are also useful for large crowd simulations, and for avoiding complicated geometry whenever possible. Another variation permitted by Maya is that of converting the image into a mesh of triangles. Seemingly convenient, the mesh then has to be cleaned up manually to gain access to the geometry of a segmented leaf. It is not obvious how one might do this without human help.

Our method of reconstruction relies on information provided by the segmentation process, such as relative height, width and offset in the original image for each image component which will tell us where to place it in the reconstructed model. All these have been written to a file in a normalized form in the previous step. This permits resizing the images after segmentation, provided the aspect ratio is preserved. This way, if rendering times are too high because of massive textures, the images can be resized and reloaded without having to change any of the scripts.

Furthermore, having information about the individual elements, they can be grouped together in a single object which makes it easy to manipulate them as a whole as well as individually. This approach has significantly improved rendering times as well as the flexibility of manipulating the reconstructed objects.

4. Applications

Static Scene Generation Although plants may seem a very narrow subject, it must be noted that they are extensively used as backgrounds in games, animations and 2D artwork. With our painted-ink models, we can generate static scenes by cloning painted plants. By editing our configuration files, we can semi-automatically generate new models by adding or removing elements from the paintings, or by combining several extracted models (Figure 11).

Animation Another possible application for our extracted paintbrush-ink is animation. For this application, we integrate our models and with Maya because that environment has a built in wind simulator. The built-in wind simulator's motion and speed parameters are keyframed for illustrative purposes in the supplementary materials.

To animate the segmented plants, they must deform un-

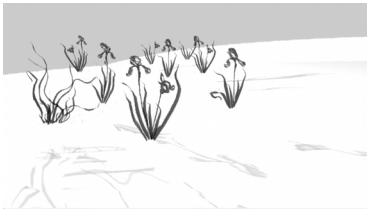


Figure 11: Generating a scene by replicating and editing the models of the ink paintings in Figure 9a.

der the influence of wind. This is achieved using the Maya representation for soft bodies. Soft bodies attach a series of particles to each vertex of the geometry. These particles then receive the effect of wind fields, and deform the original geometry accordingly. A goal-location that defines the resting position can be assigned to each particle. Goal weights control how far a particle can deviate from its assigned goal.

After the model has been constructed, our next challenge is setting up constraints that can mimic those of the real world plant. Certain plant characteristics need to be preserved, such as the unity of the object, the anchoring point, and the shape. Other aspects such as rigidity, deformation, and damping, are very difficult to estimate without prior information or training. Generally, plants can be configured following a simple rule: stiffness decreases with the distance from the anchoring point. As an example, we have implemented this rule to demonstrate our prototype.

In our prototype, we used a cosine wave with values ranging from 1 to 0 to drive stiffness. This is translated into the Maya model as goal weights. Thus, the anchoring point has stiffness 1, making it practically immobile. Toward the tip of the plant, the stiffness decreases to 0, making it more susceptible to the effects of wind. This approach achieves the goals of keeping the group structure of the individual elements by having them anchored to the root point, and of giving them some independence to the elements. During animation, the deformations of the elements are smooth, and inasmuch as the wind simulator is set realistically, the plant's motions can mimic those of real plants.

(a) Cosine wave (b) Result of using cosine heuristic

Figure 12: We used the cosine wave with values ranging from 1 to 0 as a heuristic for plant stiffness. The plant is more rigid around the anchoring point, and becomes more flexible near the tip.

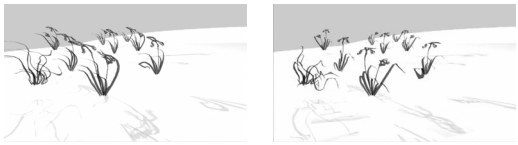


Figure 13: Snapshots from the resulting animation.

4.1. Implementation

The implementation and testing of our system has been carried out on an Intel Core 2 Duo machine with 3 GB of RAM. The segmentation stage was implemented in C++ using the OpenCV [BK08]. At 640×480 video, segmentation finishes in a matter of seconds depending on the complexity of the painted object and the number for frames in the video.

The animation stage is carried out in the Maya Framework. For this purpose, we have developed a custom Maya panel that allows the user to import plant models into a scene, customize the scene, and add wind motion to animate the plants, by choosing from a few ready keyframed wind scenarios that we provide. This plugin is implemented in the MEL interpreted scripting language. Render-times for these scenes can vary greatly with many factors. Nice lighting, good rendering quality, and high resolution require long rendering times. Also the computation of dynamics for complex scenes can also slow down the rendering process. For instance, the scene in Figure 11 containing 9 objects took 10 seconds to render at a resolution of 1280×720.

5. Testing and Results

To validate our prototype, we conducted a 12-person user study. This was necessary to confirm that the system had made successful reconstructions of the paint-stroke elements, and that the resulting animations are visually pleasing. The experiment consisted of showing the users 3 scanned paintings and 6 animations corresponding to the plants in these paintings. The users were asked to recognize which plant appears in each video (Table 1) and to give a short assessment of what they liked and disliked about the animation.

	A	B	C
A	0.84	0.0	0.16
B	0.04	0.96	0.0
C	0.08	0.0	0.92

Table 1: Table showing recognition rates of each image. Values off the main diagonal show what percent of the time the painting on that row was confused with the animated model on the column.

Some notable quotes about the positive aspects were: "looks like a painting drawn in 3D", "I liked the fact that the movement seemed quite realistic, with the tip of the leaves having a much wider movement", "simulation of leaves is

good", "most leaves provide the impression that they move naturally", "realistic animation", "looks like it's underwater", "good wind response", "shading effect is nice", "overlapping leaves darken which implies depth".

Quotes from the participants regarding shortcomings of the animations were: "when the curled up leaf bends in the wind it seems more paper-like", "the hooked leaf seemed a bit too rigid", "the ends of long leaves didn't drop down like I would expect them to", "a bit jagged at the top when leaves bend forward", "there was artifact", "animation is not physically correct", "shadow doesn't seem realistic", "base of plant immobile, doesn't interact with the wind", "the root isn't as dark as original picture", "low resolution", "some minor clipping with the floor".

The overall results of the user study were encouraging and indicate that we achieved our goals. Also, this study was valuable because we obtained independent opinions that emphasized some mistakes worth considering in the future.

6. Limitations and Future Work

Detection of pale gray tones is presently an issue. It is possible for greater parts of a painting to feature such faintly colored elements. While adjustment of the webcam's gain could help with with segmentation, that would presume our system is allowed dynamic control over the webcam. At present, we must account for the possibility that a shadow cast on the paper may have a stronger color than some drawn elements.

Our video analysis could also be improved by using explicit hand detection to be more accurate in predicting keyframes. However, getting a clear frame of a newly completed element currently has no solution. Using a setup with two cameras positioned at different angles will help, but is not a guaranteed solution either. One approach could be to train the system to recognize and associate paint-brush gestures with elements in the scanned painting.

7. Conclusions

We validated the hypothesis that a system could directly model a sumi-e artist's ink brush strokes. Further, we proposed a method for extracting a plant from a painting in the form of its semantically meaningful parts, by using the video recording of the artist at work. We have achieved a good segmentation and reconstruction of painted ink strokes by using the video recording of the artist at work, and rendered a convincing animation of the plants in the sumi-e with a new automatic reconstruction algorithm.

References

[BF03] BAUER P., FOSTER J.: *Special Edition Using Adobe Photoshop 7*. 2003. 1

[BJS*08] BARNES C., JACOBS D. E., SANDERS J., GOLDMAN D. B., RUSINKIEWICZ S., FINKELSTEIN A., AGRAWALA M.: Video puppetry: a performative interface for cutout animation. *ACM Trans. Graph. 27*, 5 (2008), 1–9. 2

[BK08] BRADSKI G., KAEHLER A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008. 5, 7

[CGZ*05] CHUANG Y.-Y., GOODMAN D. B., ZHENG K. C., CURLESS B., SALESIN D. H., SZELSKI R.: Animating pictures with stochastic motion textures. *ACM Transactions on Graphics 24*, 3 (2005), 853–860. 2

[FLJX08] FEI G., LEE W.-S., JOSLIN C., XIN Z.: 3d animation creation using space canvases for free-hand drawing. In *VRCAI '08: Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (2008), pp. 1–6. 2

[FR06] FLAGG M., REHG J. M.: Projector-guided painting. In *UIST '06* (2006), ACM, pp. 235–244. 2

[GCS02] GOOCH B., COOMBE G., SHIRLEY P.: Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), pp. 83–ff. 1

[HDK07] HORNUNG A., DEKKERS E., KOBBELT L.: Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics 26*, 1 (2007). 2

[HL94] HSU S. C., LEE I. H. H.: Drawing and animation using skeletal strokes. In *SIGGRAPH '94* (1994), pp. 109–118. 2

[HZ00] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000. 3

[KKK03] KANG S.-J., KIM S.-J., KIM C.-H.: Hardware-Accelerated Real-Time Rendering for 3D Sumi-e Painting. In *Proceedings of ICCSA 2003 (Montreal, Canada)* (2003), vol. 2669, pp. 599–608. 1

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2 (2004), 91–110. 3

[NFD07] NEUBERT B., FRANKEN T., DEUSSEN O.: Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007) 26*, 3 (2007). 2

[OOI06] OKABE M., OWADA S., IGARASHI T.: Interactive design of botanical trees using freehand sketches and example-based editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (2006), p. 18. 2

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

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling trees with a space colonization algorithm. In *Eurographics Workshop on Natural Phenomena* (2007). 2

[SXSC02] SU S. L., XU Y.-Q., SHUM H.-Y., CHEN F.: Simulating artistic brushstrokes using interval splines. In *Proceedings of the 5th International Conference on Computer Graphics and Imaging (CGIM '02)* (2002), pp. 85–90. 1, 2

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

[XXK*06] XU S., XU Y., KANG S. B., SALESIN D. H., PAN Y., SHUM H.-Y.: Animating chinese paintings through stroke-based decomposition. *ACM Trans. Graph. 25*, 2 (2006), 239–267. 2