Extracting Tepal Thickness Data from Lily Bud Cross Sections

[1]Gunnar Goetz, [1]Nicole Panek
[1]Department of Chemistry

CHEMICAL ENGINEERING
UNIVERSITY of WASHINGTON

Background

Lilies bloom by undergoing a complex combination of biological and physical processes. Differential growth strain longitudinally along the tepal cause it to buckle and bow outwards. Growth strain on the edges induce rippling along the edges of the tepals (petals). In previous work, Liang & Mahadevan studied these effects on lily blooming using computer simulations. In them, they devised a mathematical expression to describe the 3D structure of a tepal.

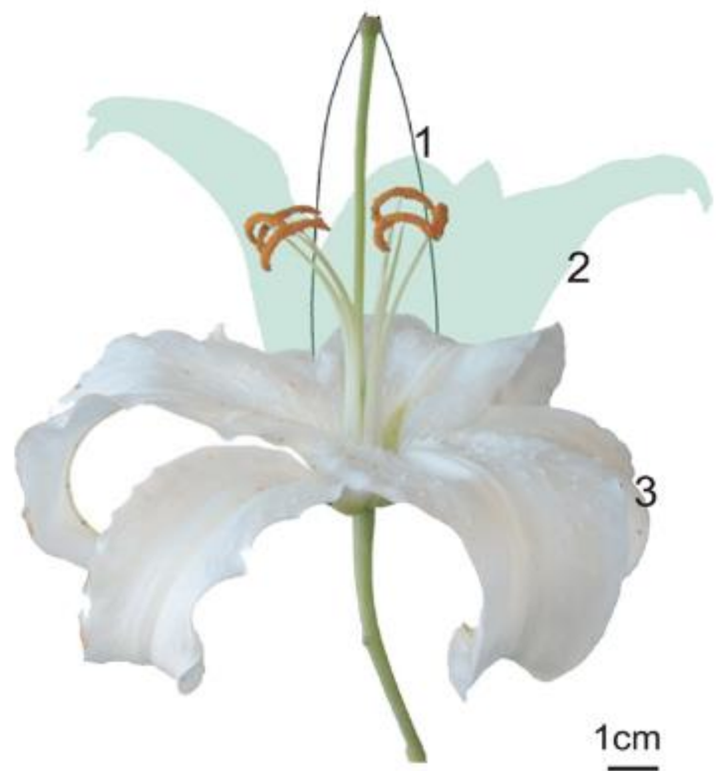

Figure 1. Stages of a blooming lily as shown in Mahadevan *PNAS*, 2011. 1 indicates a bud, 2 indicates the beginning of blooming, and 3 is fully bloomed

The equation relates maximum thickness, two parameters a and b, the semi-major and semi-minor tepal lengths respectively, and the x and y coordinate on the tepal.

$$t = t_0\left(1 - \frac{x^2}{a} - \frac{y^2}{b}\right),\ t_{lateral} = t_0\left(1 - \frac{y^2}{b}\right)$$

We'll focus on measuring $t_{lateral}$. After taking photos of lily cross sections, thickness measurements will be compared to the model proposed by Mahadevan *PNAS* 2011.

Aims

Aim 1: To convert images of lily slices into binary masks of the tepal cross-sections.
Aim 2: To take thickness measurements that are tangent to a line that bisects the tepal cross section and measures the distance from end-to-end.
Aim 3: To create a pixel-to-millimeter conversion for the scale bar in each image.
Aim 4: To compile the cross-section data and compare to the equation in Mahadevan PNAS 2011 that models lily tepals.

Methods

Our functions were written and implemented in Python using Google Colaboratory. They utilize the matplotlib, NumPy, and scikit-image libraries. The lily masks were created using the scikit-image algorithm rgb2gray and a given pixel intensity or rgb value to binarize given lily slice images. The pixel-to-millimeter conversion scale was created using the regionprops algorithm from the scikit-image measure package and the label, dilation, binary-dilation, and skeletonize algorithms from the scikit-image morphology package. Tepals were skeletonized using the skeletonize function in the morphology package. Skeleton coordinates were obtained and ordered end-to-end using the NearestNeighbors machine learning algorithm from scikit-learn, and the diff function in NumPy was used to obtain tangent vectors along the skeleton.

Results

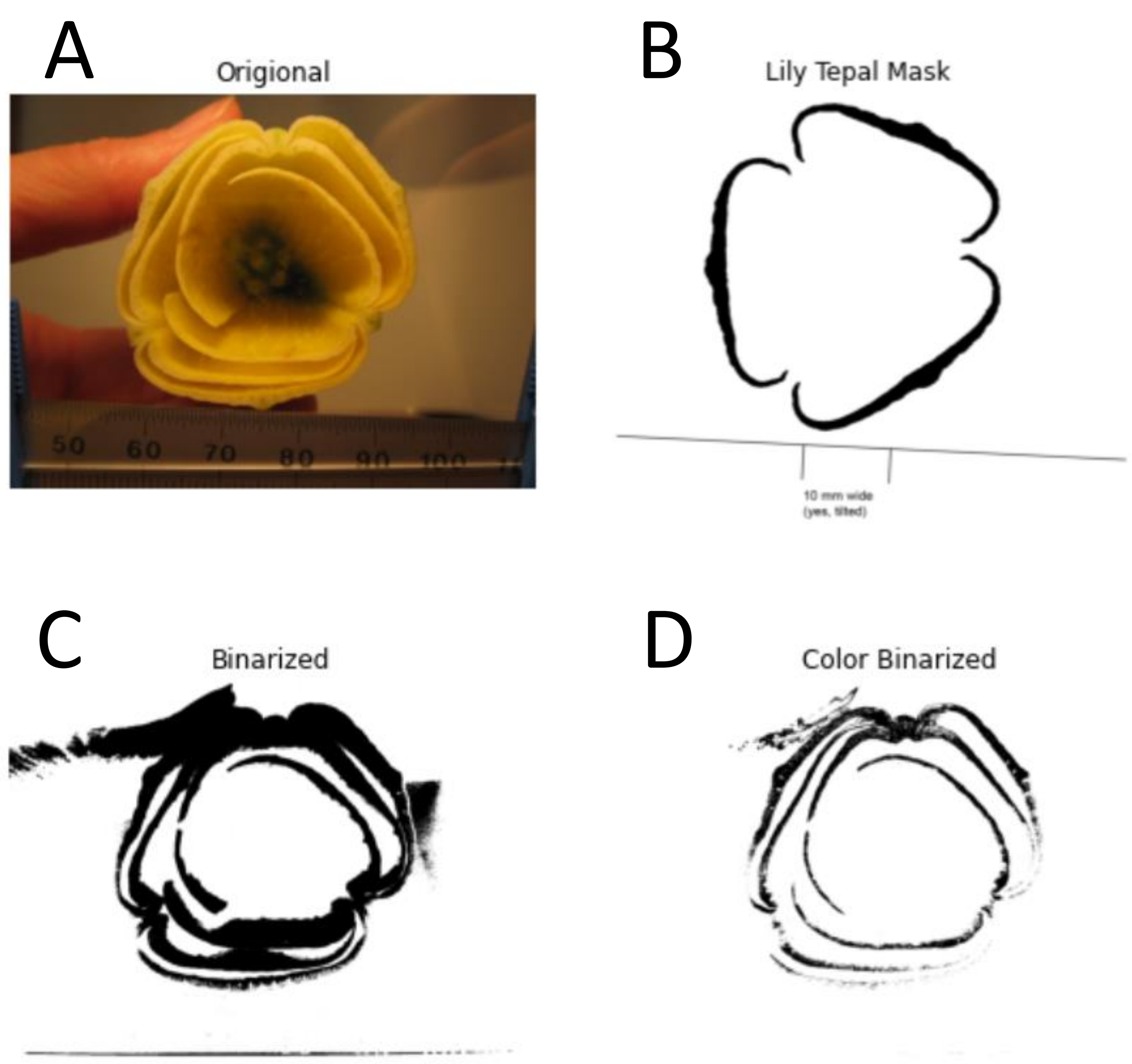

Figure 2. Example images of a lily cross-section. A) Original image taken with Canon PowerShot. B) Hand-drawn mask of the lily cross-section created with photoshop. C) Mask of the lily tepal created with our pixel-intensity based Binarize function. D) Mask of the lily cross-section created with our rgb based Binarize function.

- In the original images, the pixel intensities and rgb values for the lily tepals overlap with the intensities and rgb values of pixels in the hand holding the lily, making pixel-based binarization unreliable for obtaining masks with accurate thickness.

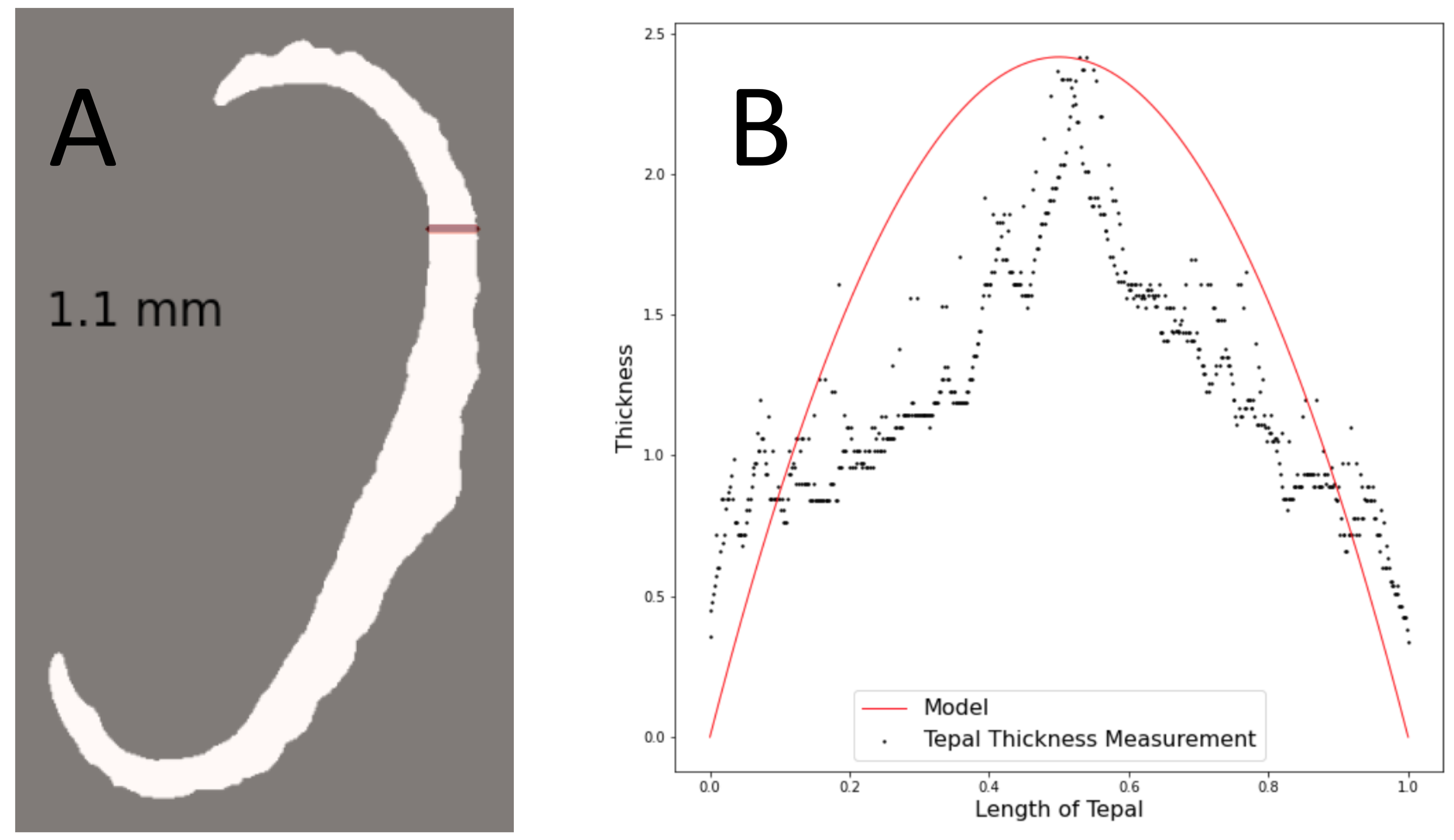

Figure 3. Measuring the thickness along the length of a cross section of a lily tepal. A) Visualization of how the thickness is measured perpendicular to one point along the skeleton of a tepal. B) A graph of the thickness data for the tepal in panel A compared to the model proposed by Mahadevan *PNAS*, 2011

- We ended up using some hand-made masks to run the thickness algorithm. In general, it was successful.
- A remaining problem is that the np.diff() function only affords 8 separate slopes which produces the large number of inflated thickness data points.

Future Work

- Write a function to more adequately calculate slope, ideally using a central difference scheme.
- Automating the process for selecting and measuring the scale bar.

References and Acknowledgements

[1] Liang, H. Mahadevan, L. PNAS, 108(14), pp. 5516-5521, 2011.
[2] Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.