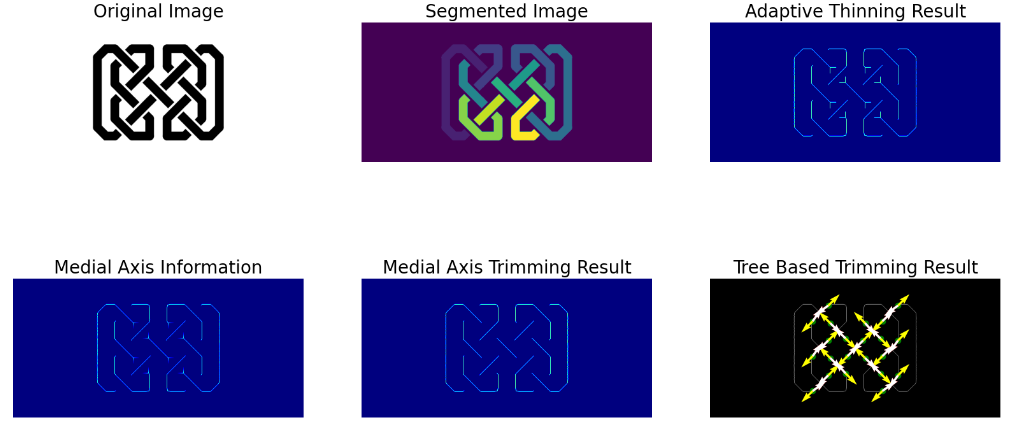
### Skeleton Trimming

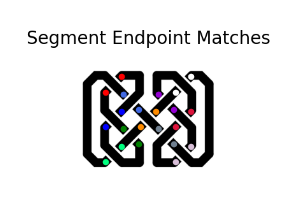
We apply two successive trimming operations, both aimed at taking us from our trimmed skeleton to a path with only 2 endpoints and no 3 way junctions.

1. **Medial Axis Trimming** The adaptive thinning method above gives us the distance from each pixel of the resulting skeleton to the closest point on the original boundary, which is called the medial axis distance. In other words, this is how much the shape had to be thinned around that pixel. Since the curve is a fixed width, the values are very centralized and thus simple thresholding a small distance below the mode yields good results in our tests. In our final version to improve robustness we start at the mode and set our threshold to be the closest standard deviation of the medial axis values that does not result in less separate segments than what we started with. This has the effect that large flanges are trimmed to be either very short or removed entirely.
2. **Tree-Based Trimming** The above trimming does not ever break up any curve but still leaves very short flanges that would otherwise mess with the interpolation of the curve. To reiterate we desire a curve with 2 endpoints and no 3 way junctions. To fix this, we use a Python library called [skan](https://github.com/jni/skan) to generate a tree representation of the pixels. After this step the image consists of a number of path elements which may or may not be a part of the same tree. First we determine what segments have a degree 3 node still and using connected components figure out what paths are a part of that segment. Then we trim the shortest constituent path for each segment that contains a degree three node. This process continues until no degree 3 nodes are left.

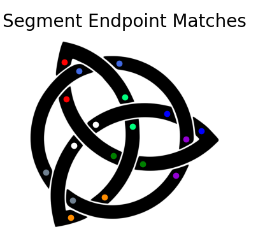
## Interpolation method

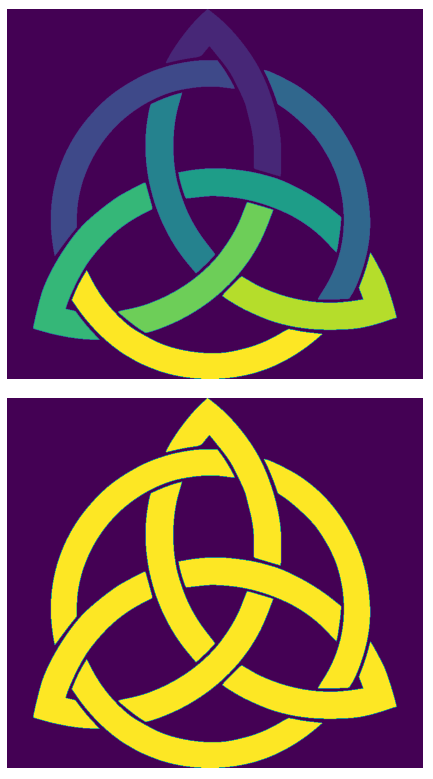
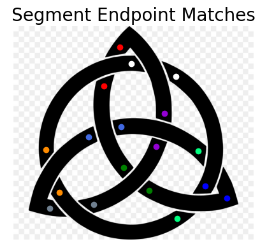
As noted earlier the path interpolation process requires a two step trimming algorithm whose results are shown here. The adaptive thinning plot shows the results after adaptive thinning but before trimming. The medial axis info plot shows the medial axis values with dimmer being lower. The tree based trimming result shows the direction of the endpoint extension used for interpolation.





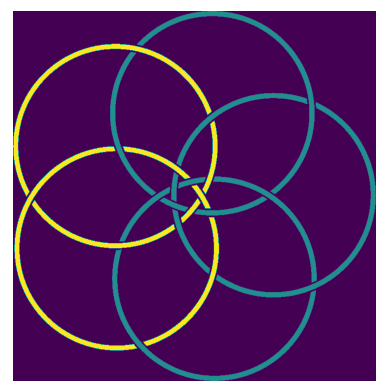
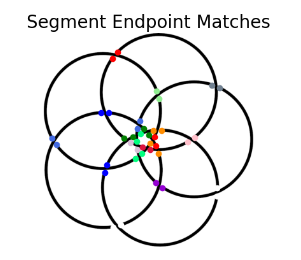
The above extensions result in the above endpoint matches which are exactly as you would expect. This method does work for a number of curved celtic knots as well. Including the trinity knot. However this method is not rotation invariant.





It has a significant tendency to fail when the endpoint of the segment going over is close to an intersection of endpoints on the underside as well and the intersection and the overlapping are closer to parallel than they are perpendicular. Since our interpolation right now does not take into account angles or extrapolate the curvature of the angle intersections like this that are close to being parallel have a high chance of swapping and when they swap between the outer and inner ring as in the above the entire knot is labeled as one strand.

With further cases such as the circle case it can be seen that this is quite a common occurrence when the the intersection is not at least close to perpendicular.



Below is another example where incorrect segment endpoint matching caused the strand identification to fail. In this case, the cause is the high density of points in the center, resulting in mismatches across circles. That results in many of the circles being labeled as part of the same strand.