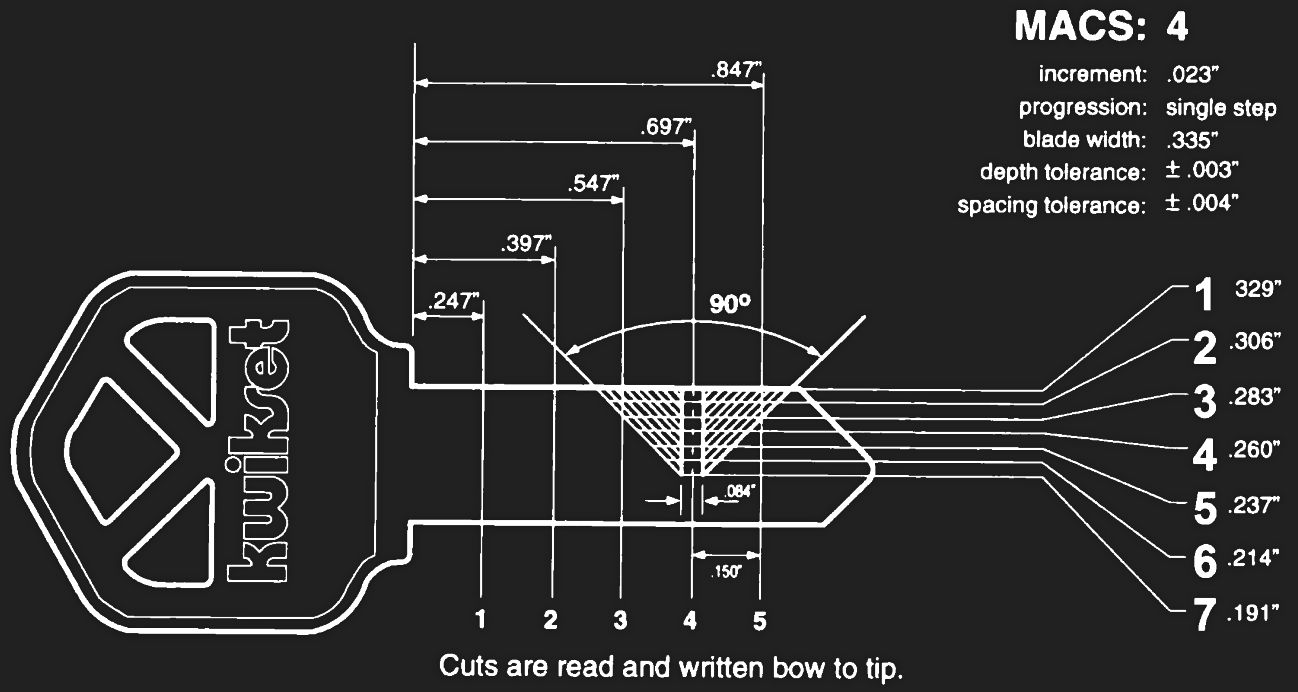
Key Decoding using Computer Vision and MATLAB

Will Freeman

Introduction – Key Bitting

- **Key bitting** refers to the cut depths of a key and can be used to recreate the key if lost.
- Each cut depth is one of several discrete values specified by the manufacturer.
- Read from shoulder to tip, these cut depths make a **bitting code**.
- This project will decode the bitting code from a photo of a key.



Theory

The following topics discussed in class will be used in this project:

- **Reading & Writing Images** – source images need to be read into MATLAB
- **Edge Detection** – edge of key needs to be detected
- **Noise Reduction using Filters** – high frequency backgrounds need to be excluded
- **Color Space Conversions** – the color of the key is not important

as well as topics not covered in this class:

- **Morphological Transformations (erosion, dilation, opening/closing)**
- **Hough Transforms**

Simulation

1. Read Image
2. Convert to Grayscale
3. Use Bilateral Filter to Smooth Image while Maintaining Edges
4. Canny Edge Detection
5. Morphological Close Operation to Connect Disconnected Edges
6. Fill All Contours
7. Remove All but the Largest Connected Component
8. Find Outer Contour
9. Use Hough Transform to find Bottom Edge
10. Rotate Image until Hough Line Horizontal
11. Use Hough Transform to Find Shoulder
12. Mark Min X Value as Tip
13. Create ROI for Key Blade
14. Find Mode of Height at Each Key Cut
15. Compare Normalized Cut Depth with Normalized Cut Specs to Obtain Bitting Code

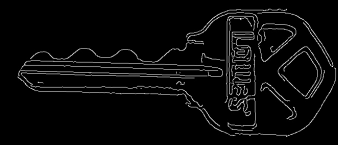
Simulation (continued)

original

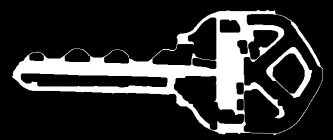
grayscale

bilateral

Canny

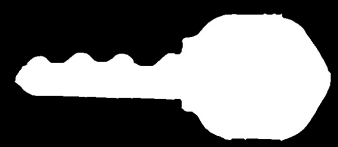


closed



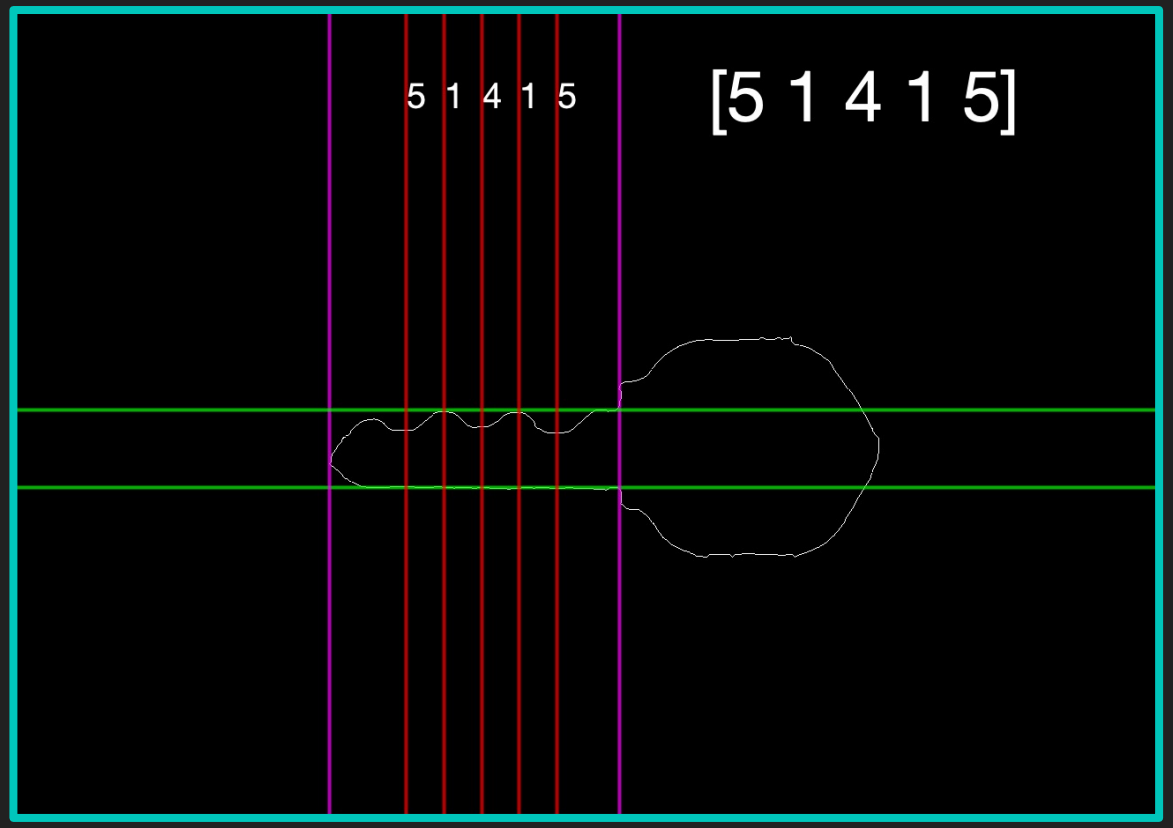
filled

largest only



contours

Simulation (continued)



- **Green Lines** – bounds of key blade
  - Bottom ← Hough transform
  - Top ← max of ROI
- **Magenta Lines** – tip to shoulder
  - Tip ← leftmost point
  - Shoulder ← Hough transform (vertical)
- **Red Lines** – location of cuts
  - normalized to blade length from specs

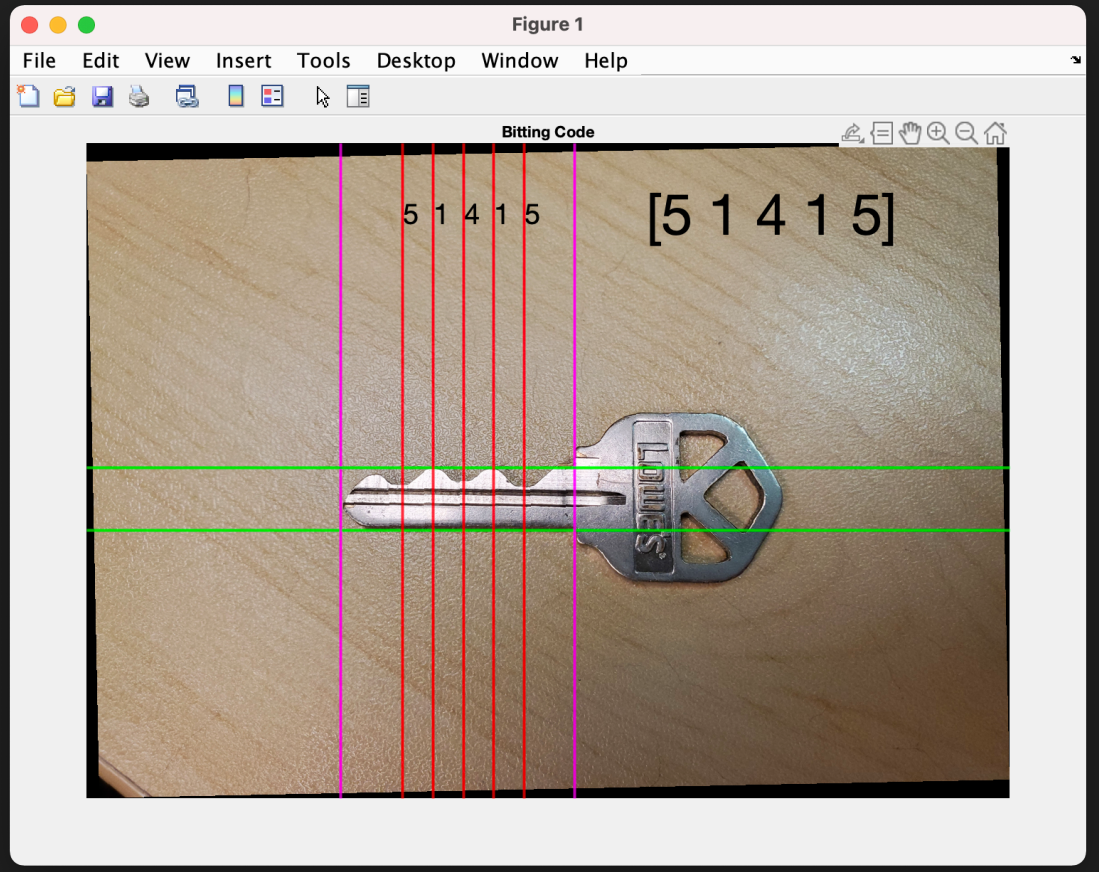
Results

Constraints

- Background should be relatively uniform
- Camera should be normal to surface
- Not many shadows present
- Key should be in this orientation

Results

- The MATLAB script was able to successfully decode this KW1 key



What Can Be Done With This?

- An attacker could take a **photo** of someone's key, and with a low level of effort, he could **decode** it.
- With the victim's **bitting code**, he could use this OpenSCAD template, created by *nrp* on Thingiverse to create a 3D model of the same key to be **3D printed**.
- 3D printed keys are very effective at opening locks and have a lifespan of a few months if taken care of.