Key Code Recognition: Case Study of Automatically Deriving the Code of a Physical Key from Mobile Device Camera Images for the EVVA A key profile

Rainhard Dieter Findling
University of Applied Sciences Upper Austria
rainhard.findling@fh-hagenberg.at

July 2, 2015

Contents

Chapter 1

Problem Definition

This technical report tackles the problem of automatically recognizing the information encoded into a physical key (key code) from a photo taken with a mobile device camera, so that a reproduction (clone) can be be created from it. Conceptually, a person takes (a) photo(s) from a key using a mobile device camera. The key code is derived from these photos using computer vision techniques and can further be used by key manufacturers to clone the key – without requiring physical access to the key sample.

1.1 Report Goal

The goal of this report is to highlight and analyze possible approaches and core methods for automatic key information derivation and recognition. For purpose of demonstrating these, a sample key photo (see figure 1.1) is used throughout the report to illustrate options and outcomes of different processing steps. The used key sample type is EVVA type A (see section 1.3), it's code is 2-1-2-5-3.

Figure 1.1: Sample photo of a physical key of 595×1000px size, from which the key code (2 1 2 5 3) should be derived.

1.2 Assumptions and Limitations

We relieve the problem by assuming certain properties when taking the key photo, which assist in successfully deriving the key code from the photo:

Photo background and contrast: we assume uniform background and good photo contrast, e. g. with the key being placed on a flat white background, such as a sheet of paper.

Illumination condition: we assume no mobile device camera flash has been used. As key surfaces easily reflects directed light and the flash is physically close to the camera, the reflected flash light will likely occlude features required in subsequent computer vision (e. g. edges or cut depth) in the photo. We instead assume the key being illuminated by a natural light source (e. g. window), with the cuts turned towards the light source and the bow to the right (as with the example in figure 1.1). With this illumination, there are no shadows cast directly next to the cuts, but only on the backside side of the key.

Photo focus: we assume good focus (a sharply focused image).

Key rotation and size: we assume the photo being taken with roughly known key position and size. The user could roughly align the key edges with a transparent line or transparent key template shown in the mobile application camera – which causes the key rotation and size to be known roughly.

Angle of photo: we expect the photo to be taken with the mobile device being helt above the key (in parallel to the key and the background), so that the angle between the camera-key-line and the key and background is close to 90° (as good as the user can do). The more different the angle is, the more it will cause image distortion, which further might decrease key recognition capabilities.

Key type: in the report we limit the analysis to a single key type (see section 1.3). We therefore assume the key type to known when deriving the key code.

In an implementation, most of the mentioned errors in taking images can be detected straight away using corresponding, separate approaches. For example, an unsharp photo, a rotated key, or can be detected, and according feedback can be given to retake the photo. An implementation can react accordingly in case an "impossible key" (impossible key properties) is detected. Depending on implementation and user interaction details, this feedback could be given on the fly (e. g. informing user that photo in unsharp and automatically taking a photo as soon focus gets sharp).

1.3 Key Properties

In this report we limit the analysis to a single key profile: EVVA type A (one of the not-closer-specified EVVA open profiles (OP) [1], sometimes also called

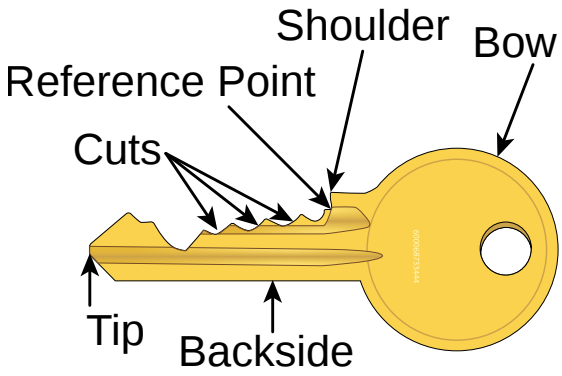


Figure 1.2: Key components and encoding information in key cuts[1].

EVVA standard profiles). This profile encodes information via cuts only (see figure 1.2).

EVVA OP in general contain either 5 or 6 such cuts, with EVVA type A containing 5 cuts. The reference point represents the height of the blank (uncut) key. It is assumed to not be affected by cutting the blank key. As typical, mechanical key cloning approaches use this reference point to clone cuts to a new key, we assume the same for our approach. As there is no detail specification available to us at the time of writing, subsequent profile details have been derived from conducted measurements (see table 1.1).

Attribute	**Measured**
Distance reference point to backside	8.65 mm
Distance shoulder to tip	27.5 mm
Amount of cuts	5
Cuts separation distance	4 mm
Distance shoulder to closest cut	5 mm
Width of key not getting cut (from reference point to tip)	2 mm
Cut width	2 mm
Amount of possible states (codes) per cut	10
Height difference of successive codes	0.5 mm
Depth of most shallow cut (code 1) to reference point	3 mm

Table 1.1: Measured EVVA type A properties.

There does not seem to be a standardized way of addressing cut heights. We define the most shallow cut code to be code 1, and the deepest cut code to be code 10. We further define the cut positions to be ordered from the tip to the shoulder, so that the cut closest to the tip is cut number 1, and the cut closest to the shoulder is cut number 5.

[1]Adapted from `https://commons.wikimedia.org/w/index.php?title=File:Parts_of_a_Yale_lock-type_key.svg&oldid=137521859`.

Chapter 2

Analysis of Applicable Approaches and Methods

The succeeding analysis is structured as step-by-step analysis of processing approaches, to obtain the key code from the given key photo (see workflow in figure 2.1). For each step[1], the problem to be solved/it's requirements are stated, then possible approaches/methods in doing so are named, and at least one is explained in detail.

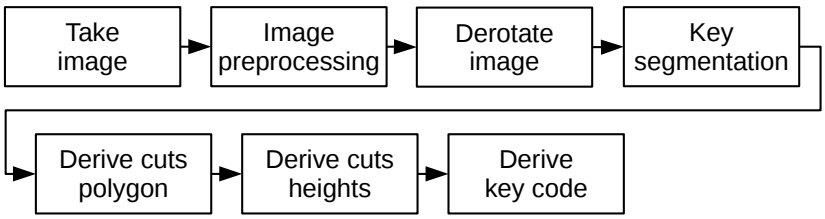


Figure 2.1: Simplified workflow of deriving the key code from a given key image (assuming the limitations of section 1.2).

2.1 Image Preprocessing

In digital image processing (DIM) and computer vision (CV), image preprocessing is frequently done before doing the actual workload [2, 3]. With image preprocessing, certain desired image properties get enhanced; e. g. increasing or normalizing image illumination or contrast. With image preprocessing, subsequent CV mechanisms (e. g. object detection or feature derivation) often is easier.

Most frequently applied proprocessing step in CV is converting the image to grayscale – as nearly all types of visual recognition tasks rely largely on structure (and less on its colorization). Further frequently used image prepro-

[1] For details and limitations of "Take image", see section 1.2

cessing mechanisms[2] [4] include normalizing image illumination of one image (e. g. with morphological operations), enhancing contrast or equalizing contrast across images (e. g. linear contrast adjustment and histogram equalization) or noise canceling to e. g. make identifying features easier (e. g. Wiener, mean or median filtering, blurring).

We propose to convert the key image to grayscale. On the one hand, in the subsequent analysis, key images without further preprocessing where sufficient to successfully derive the key code – therefore these do not seem necessary for now. On the other hand, this might be caused by certain restrictions stated in section 1.2. For this reason we suggest that further analysis (with less restrictions and multiple key samples of the same key type) takes a deeper look into the influence of image preprocessing on the key code derivation performance.

2.2 Derotating Key Images

In order to derive features from the key, which are used in further deriving the key code, the key rotation needs to be normalized first. Even if the key rotation initially is roughly known to an estimate of 1-2 degree, exact rotation in with errors below 0.5 degree is required. This is caused by some cut height measurements relying on exact knowledge of rotation – rotation offsets would cause different cut height measurement results, therefore might lead to wrong key codes.

Note that with this analysis, derotating the key is done before segmenting it. The reason for doing so is that with a uniform, monotonic background (e. g. white sheet of paper), the background can be excluded in deriving the angle of derotation, therefore influences derotation negligibly. Further, as a side effect, subsequent key segmentation is easier with known key rotation. In case the key background would be non-uniform, non-monotonic, key segmentation might need to be done before deratation. With roughly known key rotation, this would ev. include computationally expensive key template matching using sliding window principles. As the key size in the image is not known, this would further include differently size templates or a size invariant approach (e. g. Log-Polar transformation).

Derotating the key image consists of detecting edges in the image, detecting straight lines using these edges, finding the most dominant of these present lines (vertical lines of the key), deriving the average/dominant angle of those lines – then derotating the image with that angle.

2.2.1 Edge Detection

Edge detection is used to detect edges/borders in the image [2, 4]. In our key photo analysis it is a prerequisite of finding lines and polygons in the image (for polygons, see section 2.4). There exist a number of well known and frequently used edge detection approaches: the most widely used include the Canny [5] and the Sobel [6] edge detectors. Other include e. g. edge detectors of Prewitt [7], Roberts [8], Kirsch [9], Lindeberg [10] and Difference of Gaussians (DoG), Laplician of Gaussian (LoG) and zero crossing edge detection [11, 12].

[2] See e. g. `http://mathworks.com/discovery/image-enhancement.html` for practical examples.

We propose to use Canny edge detection, as is is well established and in the key photo scenario seems to produce slightly less noise than Sobel edge detection (see figure 2.2). Note that the outer right edge in both edge images is caused by the key itself, but by the key shadow. This later influences how cut heights can be measured (see section 2.5).

(a) Grayscale (b) Canny (c) Sobel

Figure 2.2: Edge detection applied on the grayscale key photo.

2.2.2 Line Fitting

Line fitting is used to detect straight lines in binary images [2, 4], such as the edge image from edge detection created in the last section. As with edge detection, there exist a number of well established approaches toward line fitting. The most well known include Hough transform (HT) [13, 14] and Random Sample Consensus (RANSAC). With HT, the quality of all possible lines in the image is evaluated, given a predefined resolution. This resolution is stated as possible line angles θ in the image – therefore, higher resolution linearly increases computational complexity. Further, in line fitting, line quality is related to the amount of positive (1) pixels a line overlaps. With RANSAC, pairs of random points are chosen to lie on a line, with all such lines getting their quality evaluated is with HT. Therefore, in contrast to HT, RANSAC is non-deterministic and represents a trade-off between result quality and computational complexity.

If computational complexity (runtime and memory) allow HT, we propose to use HT with an angular resolution of θ of $0.1°$ in the range $[-90°, 90°]$ over the complete image. Using HT, the resulting HT image is of size $2 \cdot \text{diagonal} \times ||\theta||$ (see figure 2.3), with diagonal being the edge image diagonal distance, the x axis representing the line rotation in $[-90°, 90°]$, and the y axis representing the line normal distance to the image origin in $[-\text{diagonal}, \text{diagonal}]$. Therefore, brighter pixels areas (to the outermost left and right in the image) mark parameters of lines with higher quality.

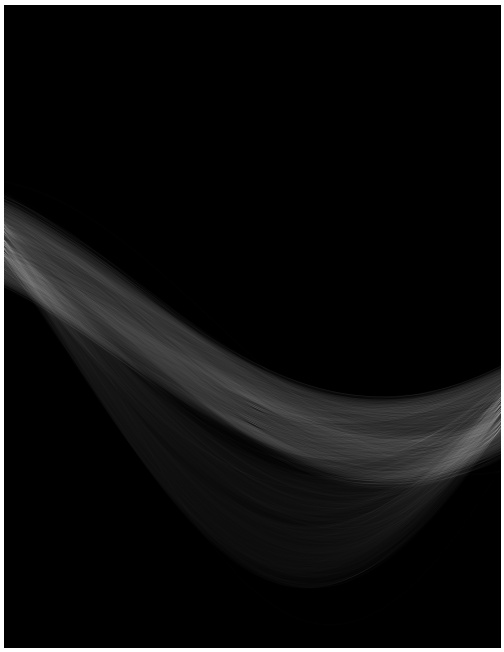


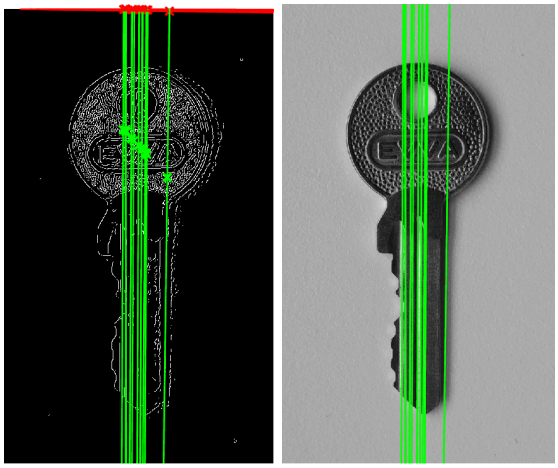
Figure 2.3: Hough transform applied on an key binary edge image. The x axis represents θ, the y axis the radius

2.2.3 Finding Derotation Angle from Dominant Lines

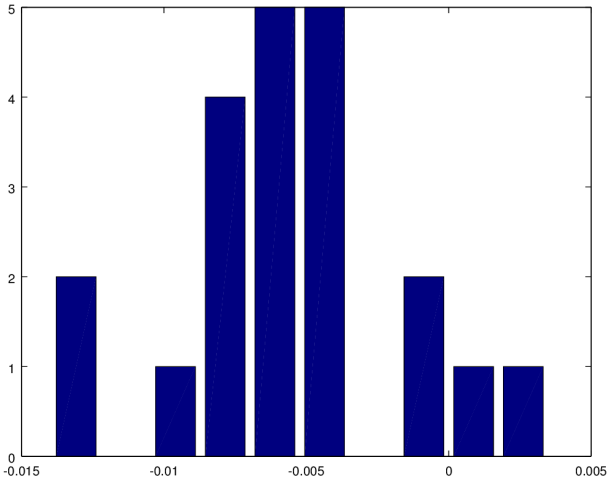
Given lines fitted to the key binary edge image, the line rotation can be used to derotate the image. It is not advised to only use a single line (e. g. the strongest line) for this, as this single line may a) be affected by line fitting errors and b) not represent the key rotation at all – which would both result in deriving wrong rotation information an wrong derotation then. An example is the line fitted to the outer right shadow of the key: although it might look parallel to the key edges an grooves, it in fact has a slight rotation offset. Using the angle of this line for derotation would result in the key featuren not being rotated as expected.

Therefore, we propose to use a more robust approach, e. g. by deriving the rotation of the most dominant lines of the image. With using multiple lines, minor random line fitting errors are likely canceling each other out, and outlier lines not representing the key rotation might be identified as such. We therefore propose to obtain the most dominant lines in relation to the single, strongest detected line – e. g. those lines with HT $\geq 80\%$ of the maximum HT detected in the image (see figure 2.4a and 2.4b). Using the rotation distribution of these most dominant lines (see figure 2.4c)[3], the key rotation can be derived using e. g. mean or median approaches. As with good edge detection and line fitting, the most dominant lines are likely associated with the key edges, we propose to use median in preference over mean – as rare outlier lines get ignored by median in contrast to mean [15], for which they will cause a rotation offset.

[3]The rotation has been normalized by π, to be centered around 0.



(a) Lines in edge image (b) Lines in original image



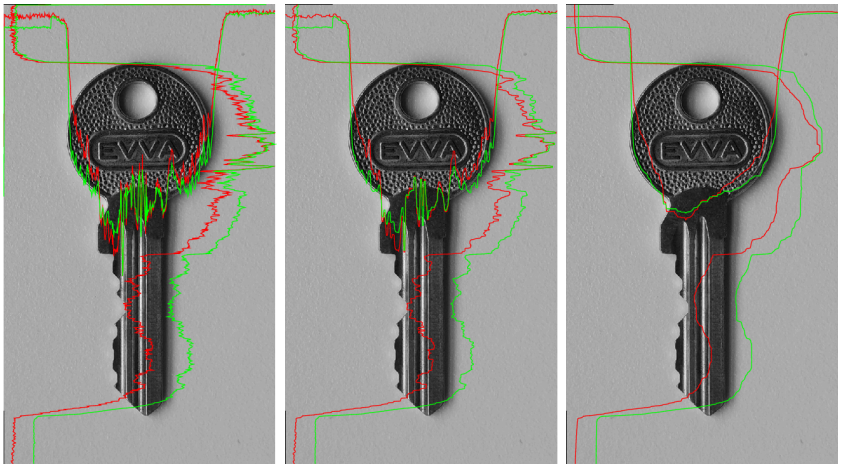
(c) Line rotation distribution

Figure 2.4: Most dominant lines fitted to the key edge image and the original image, and the rotation histogram of these most dominant lines, from which the derotation angle is derived.

2.3 Key Segmentation

Using the derotated key image, key segmentation (cropping to areas containing key information) can in done as next step. Segmentation is necessary to i.a. obtain key borders (e. g. the tip). Given a monotonic and uniform background, the deviation per rows an columns can be used do determine the start and end of key related image parts. Given a non-monotonic and non-uniform background of the key photo, a more sophisticated and computationally more expensive approach may be required. Such an approach might include sliding window matching of a key template in different sizes and with increasing resolution of matching steps.

For deviation based determination of the key region, e. g. standard deviation (SD) or the statistically robust median absolute distance (MAD) can be applied per row and column [16]. To smooth the deviation course, usually a sliding window filter [17] of window size w_s is applied, calculating the moving average or moving median of the sliding window (see figure 2.5).



(a) Original

(b) Median sliding window, $w_s = 5$px

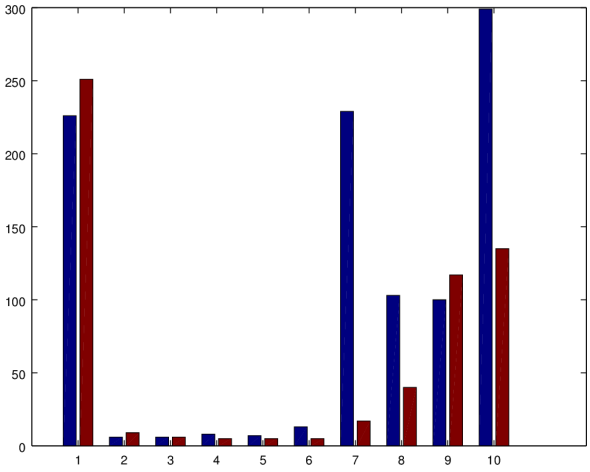
(c) Median sliding window, $w_s = 100$px

Figure 2.5: Deviation levels computed from SD (red) and MAD (green).

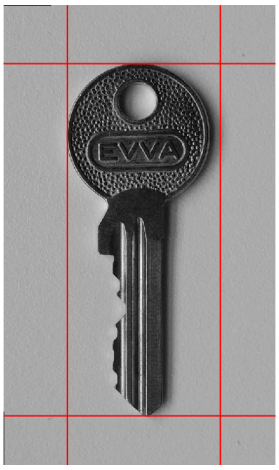
To obtain the key region of the image we propose to use a deviation threshold per row and column. The first and last deviation value of a row or column above that threshold mark the start and end of the key related error. The threshold can be related to the max deviation detected over all rows and columns – e. g. 20% of the maximum deviation found (see distribution of deviation in figure 2.6a and detected key region from deviation thresholding in figure 2.6b).

2.4 Deriving Cuts Polygon

After derotating and segmenting the key area of the key image, derivation of information related to the key code can be started. The first step in doing so is to derive the polygon of the cuts (outer border on the cuts-side of the key),



(a) Deviation distribution



(b) Key region from deviation thresholding

Figure 2.6: Deviation distribution horizontal (blue) and vertical (red) from low (1) to high (10), and key region to be cropped, derived from deviation thresholding with 20% of maximum determined deviation.

which represents the information encoded into the key. Using this polygon, the cut heights and codes can be derived later (see section 2.5 and 2.6).

2.4.1 Polygon Derivation

An approach to derive the cuts polygon (outer border of the key) is to a) again apply an edge detector (see section 2.2.1, in our example we use the Canny and Sobel edge detectors again) to obtain an derotated and segmented, binary edge image of the key, then b) add the first/last (outermost left/right) pixel per row that indicates an edge to the cuts polygon. This approach requires the cuts to be (almost) free of shadows, hence the key has been rotated to face the light source during recording. Other approaches that could be used to derive the cuts polygon include e. g. Gradient Vector Flow snakes [18], which could be applied on an edge image as well (but as these approaches come at the cost of higher computation complexity, we use the simpler method of obtaining the polygon row-based in this approach).

The hereby obtained polygon covers the key from start of the bow to the tip (see figure 2.7a). As edge detection is noisy, the hereby obtained polygon is noisy as well. Again, sliding window filtering (mean, median) can be used to smoothen the polygon. As the Sobel based derivation seem slightly noisier, we propose to again use the Canny edge detector as basis for this step. For smoothing we propose to not use mean but median filtering. Mean filtering will cause smoothed cuts polygon maximas/minimas and will be affected by outliers in the form of an offset. In contrast, median filtering ignore outliers [15] (if the are rare enough), therefore has less tendency to smooth cuts polygon maximas/minimas.

(a) Derived cuts polygon

Figure 2.7: Unfiltered derived cuts polygon based on a Canny (red) and Sobel (green) binary edge image of the key.

2.4.2 Shoulder Determination

Given the derotated cuts polygon of a key, the shoulder position needs to be determined. A number of approaches can be used to determine the shoulder position in the polygon, such as template matching [19] (as the expected polygon form is known), or determining the third polygon peak/first straight/steady region of the polygon. Another, simpler approach includes determining the polygon's strongest declination – as the shoulder involves the single strongest, nearly horizontal/vertical cut. Therefore, the derivation of the median-smoothed cuts polygon is calculated. The maximum declination in this derivation marks the shoulder position (see figure 2.8a). This position can be used to crop the cuts polygon from the shoulder position to the tip – which has previously been found in key segmentation (see figure 2.8b and 2.8c).

This cut polygon can now be used to either do a 1:1 cuts copy of the key (as it is done manually by certain manufacturers and key cutting machines), or it can be used to derive the cut heights and the key code (requiring exact knowledge about these features).

2.5 Deriving Cut Heights

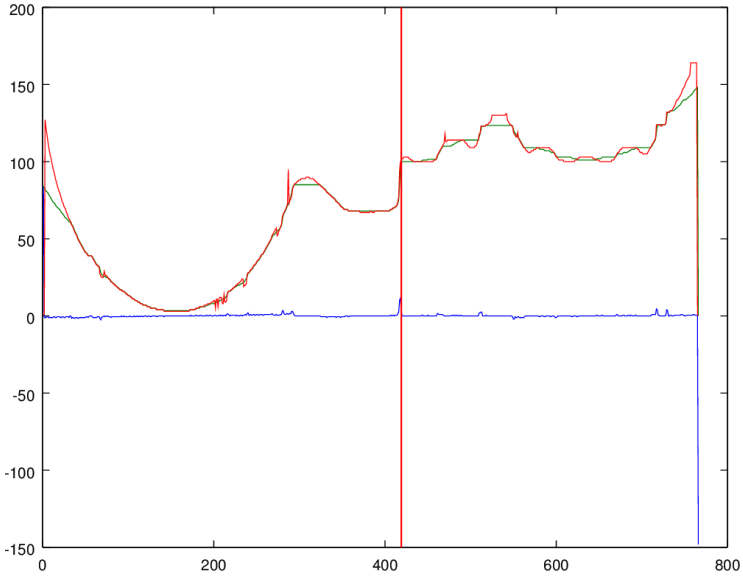
After obtaining the polygon covering the cuts only, the cuts ("feature") heights can be determined. This requires a) the cuts positions to be known in relation to the shoulder and tip, and b) a reference height to robustly measure cut heightss against.

Cuts Positions

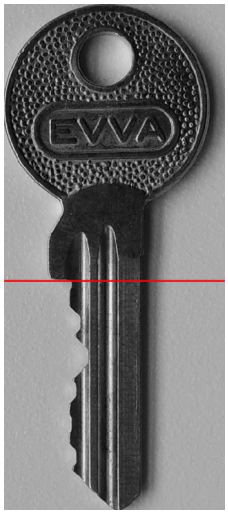
The cuts positions vary depending on the key type. Consequently, measuring the cut height for different key types requires knowledge of the corresponding cuts positions of the given type in the form of a database (besides requiring a separate mechanism to determine the key type in the first place). For our example key type EVVA type A, the 5 cuts positions are defined as a distance d in mm from the shoulder, with $d = 5 + 4 \cdot c_i$, and the cut index $c_i = [0, 3]$, at which cuts are centered and height measurements take place.

Cut Heights

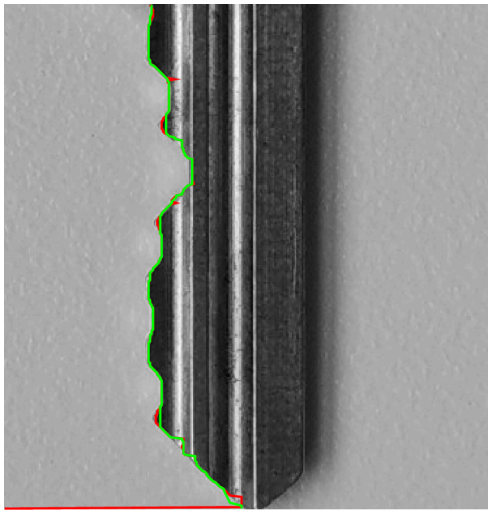
For measuring the cut heights, a reference height is required. Using the backside (upper side) border of the key would seem intuitive – but as shadows cast by the key might be recognized as edges and could be mistreated as the key backside edge, it is difficult to reliably use it as reference height. A number of other potential reference heights are affected by similar problems: e. g. the key grooves will be illuminated differently by different angles of light, therefore will be detected as slightly different positions and cannot easily be used as reliable reference height. Therefore, we propose to use a different reference height for cut heights measurements. We define the height h_0 of the region of polygon right after the shoulder to be the reference height (see figure 2.9). Theoretically, this region is of size 2-3mm and represents the original height of the blank and uncut key. Although, as this region is not required for the locking mechanisms to work



(a) The strongest declination in the derivation (blue) of the unfiltered (red) and filtered (green) cuts polygon marks the shoulder position (red line).



(b) Shoulder position



(c) shoulder segmented cuts polygon, filtered (green) and unfiltered (red).

Figure 2.8: Deriving the shoulder position and cropping the cuts polygon to the cuts region.

correctly, there might exist key samples that have different heights/erroneous cuts here – but at the time of writing, we are not aware of any such samples.

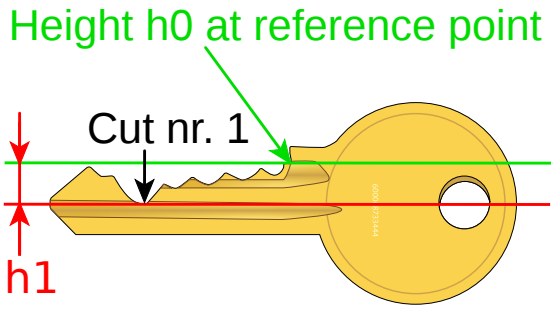


Figure 2.9: Cut heights h_n are measured relative to the reference height h_0[4].

Using h_0 as reference height requires precise derotation: changes in key rotation causes bigger offset the farther the cut lies apart the reference height. e. g. for the cut closest to the tip, which is in about 20mm distance to h_0, a rotation around h_0 of 1° would result in about 0.35mm offset in measured cut height – which likely causes wrong subsequent code detection (as code heights are only 0.5mm apart). Further, cut height can be measured in two ways: using a single height measurement in reference to h_0, or measuring the height of a narrow region that is associated with a cut. As the cuts polygon might be erroneous on single height measurements, and as cuts intentionally have a width of about 2 mm, a 2 mm sliding window for smoothing height measurements seems promising. Here we again propose to use median over mean filtering, as median filtering is more robust to outliers. Figure 2.10 shows the measured cut heights, marked at the predefined cuts positions. The cut height seem robustly detected. These cut heights can now be used in the final step of deriving the key code.

2.6 Deriving the Key Code

The key code can be obtained in multiple ways from previous processing results. One option is to treat the cuts polygon or its derivation as signal and do a classification/similarity search in this cuts-polygon-space. The cuts polygon space could be generated from knowing all possible key cuts polygons, which have been generatey artifically, and which have been preprocessed using the same approaches as the detected polygon (median filtering etc.). Polygon comparison could be done e. g. by signal processing approaches, which the correlation coefficient (e. g. Pearson [20], Spearman [21] or Kendall [22]) being one of the most frequently used approaches in doing so.

Having obtained the cut heights previously, an easier way to obtain the key code is to directly decide on the most likely code for each single cut from it's height – given all possible cut heights and their respective codes (see equation 2.1, with c_n being the n-th detected code, h_n being the measured height of

[4]Adapted from `https://commons.wikimedia.org/w/index.php?title=File:Parts_of_a_Yale_lock-type_key.svg&oldid=137521859`.

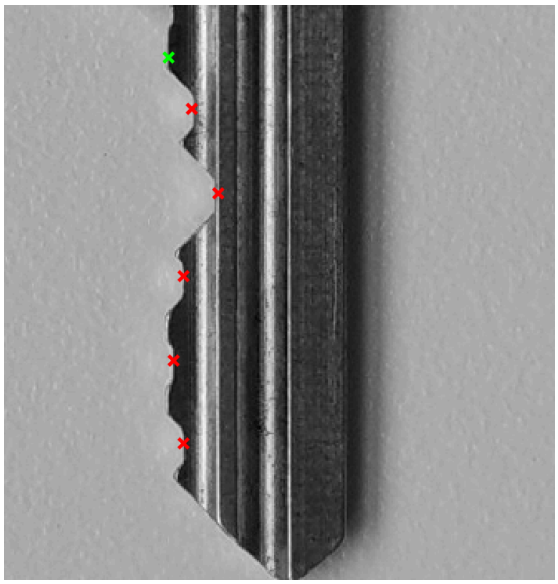


Figure 2.10: Reference height h_0 (green) and measured cut heights relative to h_0 (red) using 2 mm median cuts polygon filtering, marked at the predefined cuts positions.

the n-th cut, h_b being the base code height, which is the height of the highest code in relation to h_0, and h_d being the height difference between codes).

$$c_n = 1 + \frac{\text{round}(h_n - h_d)}{h_c} \tag{2.1}$$

Using this approach we have been able to successfully derive the correct code 2-1-2-5-3 from the key photo sample.

Chapter 3

Conclusion

In this case study we derived key code information from mobile device camera images of EVVA type A keys. To do so, we manually combined digital image processing and computer vision techniques in serial steps applied to the key photo, to normalize key rotation, do key segmentation (cropping of the photo to areas relating to key information), derive the key cuts polygon and finally the key code from the cuts polygon. Using this approach we could successfully derive the correct key code for the available key sample. We therefore infer that deriving the definite key information (e. g. required to clone a key) is possible at least for certain types of keys using computer vision approaches and mobile devices, and suggest to further investigate this case.

Possible extensions and next steps to our investigation include: a) investigating the automation and parametrization of the steps we proposed to derive key information from key photos and b) a more extensive evaluation of deriving the key code results in finding the correct code (featuring multiple key samples with different key codes). Future research could further investigate the effects of different illumination conditions and using different mobile device cameras and on deriving key information.

Bibliography

[1] EVVA Sicherheitstechnologie GmbH, "Product catalogue: mechanical locking system eps and conventional locking systems," online, Nov. 2010. [Online]. Available: http://www.evva.at/fileadmin/files_all/Download/Produktkatalog/Product_Catalogue_EPS.pdf

[2] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[3] G. Stockman and L. G. Shapiro, *Computer Vision*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[4] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, Sep. 2010.

[5] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[6] I. Sobel. (2014, Feb.) History and definition of the so-called "sobel operator". online. [Online]. Available: http://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator

[7] B. Lipkin, *Picture Processing and Psychopictorics*. Elsevier Science, 1970. [Online]. Available: https://books.google.de/books?id=vp-w_pC9JBAC

[8] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology. Dept. of Electrical Engineering, 1963.

[9] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, vol. 4, no. 3, pp. 315–328, 1971. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0010480971900346

[10] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–156, 1998. [Online]. Available: http://dx.doi.org/10.1023/A%3A1008097225773

[11] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[12] R. Haralick, "Digital step edges from zero crossing of second directional derivatives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, no. 1, pp. 58–68, Jan. 1984.

[13] D. H. Ballard, "Readings in computer vision: Issues, problems, principles, and paradigms," M. A. Fischler and O. Firschein, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987, ch. Generalizing the Hough Transform to Detect Arbitrary Shapes, pp. 714–725. [Online]. Available: http://dl.acm.org/citation.cfm?id=33517.33574

[14] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972. [Online]. Available: http://doi.acm.org/10.1145/361237.361242

[15] A. Hayter, *Probability and Statistics for Engineers and Scientists*. Cengage Learning, 2012. [Online]. Available: https://books.google.at/books?id=Z3lr7UHceYEC

[16] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th ed. Chapman & Hall/CRC, 2007.

[17] G. Arce, *Nonlinear Signal Processing: A Statistical Approach*, ser. Wiley InterScience online books. Wiley, 2005. [Online]. Available: https://books.google.at/books?id=4DKZH0aPXMMC

[18] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *Image Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 359–369, 1998.

[19] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing, 2009.

[20] K. Pearson, "Note on regression and inheritance in the case of two parents," in *Proceedings of the Royal Society of London*, vol. 58, Jan. 1895, pp. 240–242.

[21] C. Spearman, "The proof and measurement of association between two rings," *American Journal of Psychology*, vol. 15, pp. 72–101, 1904.

[22] M. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, Jun. 1938.