Final Project: Campus Card Recognition
Comp4102 A Computer Vision

Jiahe Geng
Student Number:101056037
jiahegeng@cmail.carleton.ca

Jiacheng Tang
Student Number:101038546
jiachengtang@cmail.carleton.ca

Yuhua Chen
Student Number:101035484
yuhuachen@cmail.carleton.ca

Abstract

The deliverable of the project is a student information management system, which could scan and extract the information on the student card and compare it with the information in the database to quickly find the student information. The system will also display all the student information (i.e. first name, last name, department, total credit, etc.). What is more, the user could delete, update or add student information to the database. For the challenges of the project, even though the system is implemented by using some pre-existing functions in OpenCV and MySQL, but it still has to be able to recognize the object with a different and complicated environment and handle some problems with the picture such as glare in the photo or wrong direction of the photo.

1 Introduction

The project is focusing on solving the campus card recognition problem with the preposition camera of the personal computing device. The software will turn the camera of the device, capture the frame of the campus card, and obtain the student's information from the taken image.

The recognition process includes matching features to compute a homography of campus card, locating the student number, and digits recognition. After obtaining the information from the campus card, the program would try to match the student number from the database. If the verification is successful, the program will display the detailed information of this student. Otherwise, if there is no information matched, the application will ask the user to enter the new student's data to the database.

2 Background

2.1 OpenCV

OpenCV package (Open Source Computer Vision Library: `http://opencv.org`) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. Most of the image processing algorithms and methods in this project are stemming from this package. [1]

2.1.1 Find Contours

According to Suzuki, S., Be, K. (1985),in this paper, the author analyzes a method to the topological structure of gray-scale images through border following which is an extension of the border following algorithm which discriminates between the outer borders and the hole borders. All of these modified versions are effective and quick. [2]

2.1.2 Template matching

The cv2.matchTemplate() computes a proximity map for a raster template and an image where the template is searched for. What is more, this function is the core of the recognition since it would build a map between the digit contours on the card and the reference image, which would be explained in the approach part.

2.2 Imutils

Imutils package is a series of convenience functions to make basic image processing functions such as trans-

lation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python. It enhanced the functionality of the openCV, and allowed this project to have a ability to sort the contours by location and reshape the digit shapes.

2.3 Relevant Papers

Peng, C. (2015). Position-weighted template matching for measuring in-plane dynamics of microdevices (T) provides the idea and the algorithm for matching the element in the specific image and template. [4]

Suzuki, S., & Be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics and Image Processing provides two border following algorithms are proposed for the topological analysis of digitized binary images. Since that, these algorithms can be effectively used in component counting, shrinking, and topological structural analysis of binary images, when a sequential digital computer is used.[3]

3 Proposed Approach

3.1 Image Capture

The project allows the user to turn on the front camera and capture the image of their campus card. The function cv2.VideoCapture(0) is required to launch the camera of the device and start the live stream. To capture a single image, the user needs to place their campus card in the specified rectangle space in the live stream. These could help to minimize the calculation of searching the campus card in the entire image and reduce the processing time. The program also expected the user to press a specific keystroke after the campus card has placed in the correct spot and save a particular frame from the live stream.

3.2 Identify card edges

Although when the user captures the campus card image, the program has specified a rectangle space for the user to place their cards. The campus card is not always perfectly fitted in the rectangle area, and any displacement may cause the failure of the number locating. The perspective transform is still required in the preprocessing step.

Firstly, we planned to use the document scanner method to find the edges of the campus card and apply the perspective transform. We find contours in our edged image, use the function, cv2.approxPolyDP(), to obtain contours that approximate to the shape of a quadrilateral, retaining the contour that has the area and the perimeter which are higher than the specific value, and discard the rest of the contours. However, this method does not demonstrate excellent performance in finding the contours of the campus cards. The shade, background noise all have a severe impact on detecting the edges of the campus cards, especially when the user holds the campus card in hand and scanning the campus card using the front camera.

The Image Alignment Algorithm provided a great solution to those problems. Given images A and B, compute the image feature for A and B, then match the features between A and B. Lastly, compute the homography between A and B on sets of matches.

In the project, we first find SURF features in the campus card image and the two reference images shown in Figure 1. SURF, which stands for Speeded Up Robust Features, it is a speeded-up version of SIFT. SIFT approximated Laplacian of Gaussian with Difference of Gaussian for finding scale-space, but SURF approximates LoG with Box Filter. The below image shows a demonstration of such an approximation. SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.

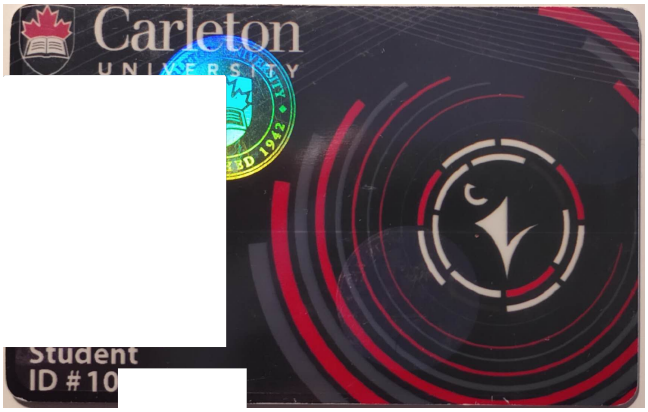

Fig. 1. The ref for feature matching.

FLANN based Matcher(which stands for Fast Library for Approximate Nearest Neighbors) is used to find matches between images. It contains a collection of algorithms optimized for fast nearest neighbour search in large datasets and high dimensional features. It works faster than BFMatcher for large datasets.

After matched features, we use cv2.findHomography() to compute homography on the set of matches. The function finds and returns

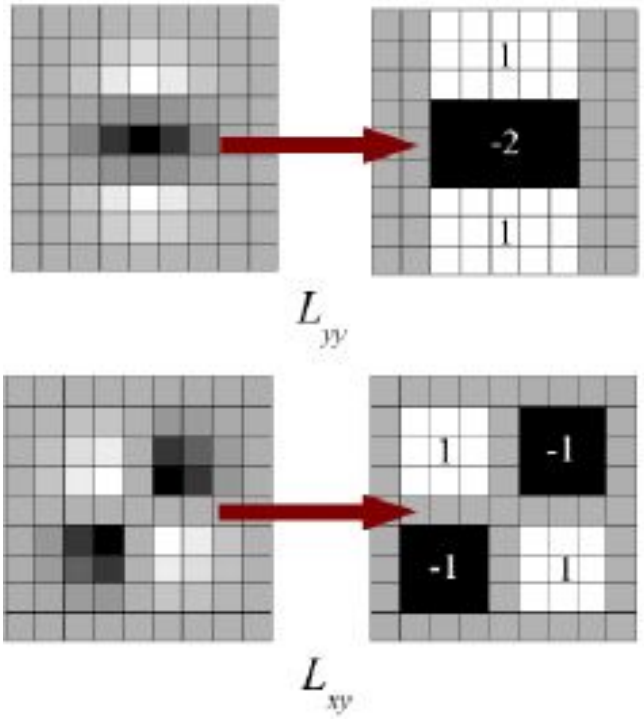

Fig. 2. SURF box filter

the perspective transformation between the source and the destination planes. So that the back-projection error

$$\begin{bmatrix} ax_i' \\ ay_i' \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Fig. 3. Solving for homographies

is minimized.

$$\sum_i (x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}})^2 + (y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}})^2$$

3.3 Card number locate

In this project, the method of "findContours" is used to located the numbers and imutils library which is a series of convenience functions to make basic image processing operations such as translation, rotation, resizing and skeletonization to enhance the functionality. "findContours" is a function of finding contours in a binary image. The function retrieves contours from the binary image using the algorithm about the properties of For an arbitrary 1-component of a binary picture, its outer border is one and unique. Generally, for any hole, its hole border (the border between that hole and the 1-component which surrounds it directly) is also unique, which is expained as the Figure 4. The contours are useful for shape analysis and object detection and recognition. [1] Basically, the algorithm

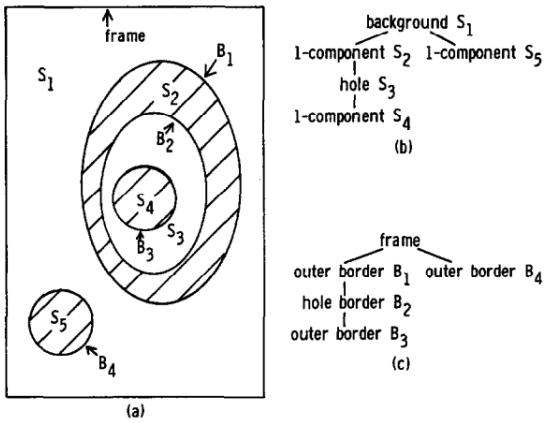

Fig. 4. Surroundness among connected components (b) and among borders (c).

which is proofed by Suzuki, S. and Abe, K.(1985)[1], is based on the Green's theorem. In mathematics, Green's theorem gives the relationship between a line integral around a simple closed curve C and a double integral over the plane region D bounded by C.

$$\oint_C (L\,dx + M\,dy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) dx\,dy \quad (1)$$

The specific steps: First, find the contour which is the area of the student number. Then, build a window for this area to repeat the previous steps using a loop until finding the nine contours for the nine digits.

3.4 Digits recognition

Before the main part of identification, we identified the font of the student number on the campus card, and the best match type is "Arial". So, the Figure 5 is the specific reference to recognize the digit. "matchTemplate" is the main method to achieve the identification of student number and it is a kernel tracking approach that locates regions of an image that match a predetermined template.The steps of the template matching

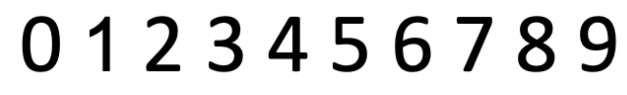

Fig. 5. The reference digits for recognition.

methods in OpenCV can be summed as follows from Peng, C. (2015)[4]:
1) Load an input image and an image template.
2) Implement a template matching procedure by using Normalized Cross Coefficient Template Matching method.
3) Normalize the output of the matching procedure.
4) Determine the location with the best match. The lowest value in R indicates the best match when using Squared Difference Template Matching metric and its normalized version. However, for the other four metrics, the higher the value, the better the match.
5) Draw a rectangle around the area corresponding to the highest match
A reliable method for exploring and discovering the location of a template image in a larger image is "Template Matching". OpenCV comes with a function cv2.matchTemplate() for this purpose. The algorithm slides the template and patch of input image under the template image with several comparison methods. Finally, it returns a gray-scale image whose pixels indicate the level that the neighbourhood of that pixel match with template.

For this target, the method with "CV_TM_CCOEFF_NORMED"(Normalized Cross Coefficient Template Matching) has the best performance which is according to the formula:

$$R(x,y) = \frac{\Sigma_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\Sigma_{x',y'}(T'(x',y')^2 \cdot \Sigma_{x',y'} I'(x+x',y+y')^2)}}$$

The output image will have a size of (W-w+1, H-h+1) when the input image is of size (W * H) and the template image is of size (w * h). Once getting the result, the program can use cv2.minMaxLoc() function to find the location of maximum/minimum value. After that, take it as the top-left corner of the rectangle and take (w,h) as the width and height of the rectangle. Then the rectangle is the region of the template from Peng, C. (2015)[4].
What is more, the device limitation could such as resolution ratio lead to some unexpected issue, which is the ambiguous and spurious boundary error.
Sharpen filter can emphasize details and enhance the edges of objects in the image, which is critical when post-processing many types of images.
The sharpening process works by creating a slightly blurred image, the unsharp mask. Then, subtract this from the original image to detect the presence of edges. Then use this mask to selectively increase the contrast along these edges, leaving a sharper final image.
The following is a Typical kernel for sharpening:

$$\text{Kernel for sharpening} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2)$$

3.5 Database

The database of this project is a table that contains a record for each student. This called the student table, and it contains each student's student number, first name, last name, department, total credit. Each row could uniquely represent a student. Because the student number is unique for each student, the student number is the primary key for the table. After scanning the student card, the system would call cursor() to create a cursor object, and use execute() to let MySQL execute the operations we specified, such as insert, remove or update. When MySQL finishes the work, the system will call fetchall() to receive the result. Here is the example for the database relation instance as Figure 6

id	firstname	lastname	dept_name	tot_cred
101035484	Yuhua	Chen	Comp. Sci.	37
101038546	Jiacheng	Tang	Comp. Sci.	35
101056037	Jiahe	Geng	Comp. Sci.	40
101112772	Haoyang	Wang	Bio. Sci.	5
11111111	Lily	Tim	Bis	37
14105444	jasper	Tang	Arts	5

Fig. 6. The example of relation instance.

4 Results

In our project, we allow the user to upload a local image of the campus card or take a photo of the campus card using the front camera. However, due to the limitation of device functionality, such as the resolution of

the personal computer front camera, the performance of recognizing student number with computer camera have lower accuracy.

4.1 Homography transform and Locating card number

As shown in the Figure 7 and 8, by using the reference image shown in Figure 1, the campus card image with different scales, background and resolution can compute homography successfully.

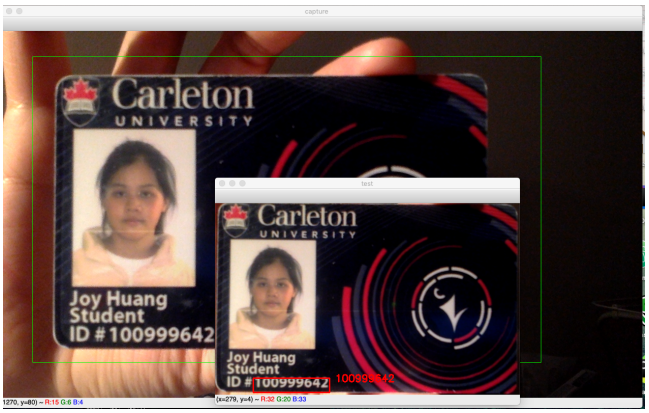

Fig. 7. homography transform of Image capture by front camera

Fig. 8. homography transform of local Image capture by phone

After the homography transformation, the student number region will always lie on the same spot.

4.2 Recognizing student number

If the recognition successful, the detected student number will be drawn on the image as shown in Figure 9 The program then uses that student number to find the match in the database. If the student number found in the database, the application will display the student information shown in Figure 10. The user could also select to update or delete the student's instance from the database.

If the student number detected but the corresponding student's data cannot found in the database, the sys-

Fig. 9. The result of digit recognition.

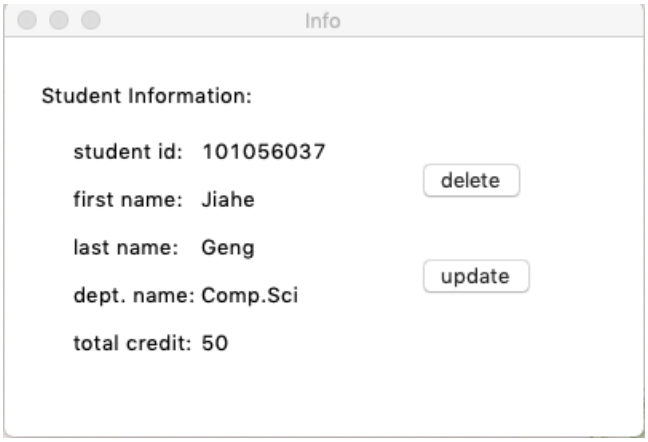

Fig. 10. Found matched student number in database

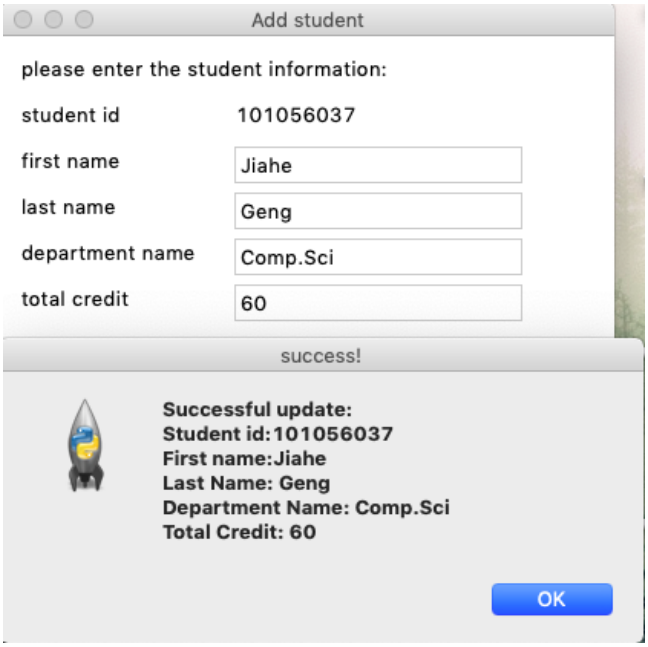

Fig. 11. Update student number in database

tem would notify the user that the student id not exist, as shown in Figure 12

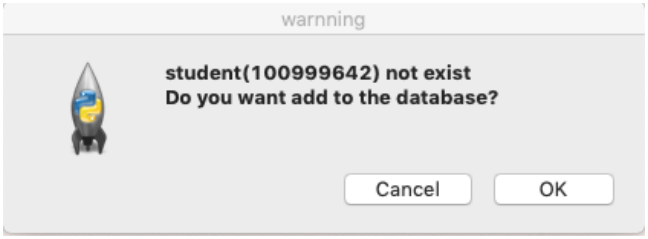

Fig. 12. student number not exist in database

4.3 Failure cases

Several cases might cause the application to fail to recognize the student number.

a) In the design of the Carleton Campus Card, the spacing between each digit is inconsistent, especially between the digit 4 and the next digit. As shown in Figure 13, the spacing between digit 4 and 8 are closer than other digits, which result in the application fail to detect digit 8 in some cases. Furthermore, as shown in Figure

Fig. 13. The result of Failure.

14, the position of the first digit not always consistent when we align the campus card using the location of "ID".

b) Some Campus cards contain abrasion that is in white, which is hard for software to distinguish the student number and abrasion. As the example shown in Figure 15, a white abrasion appears under the digit 7, which cause the system to analyze the digit 7 as 2.

c) As mentioned above, Even though we try many method to sharpen and enhance the image, the resolution of the computer is the major problem that causes the failure of recognition

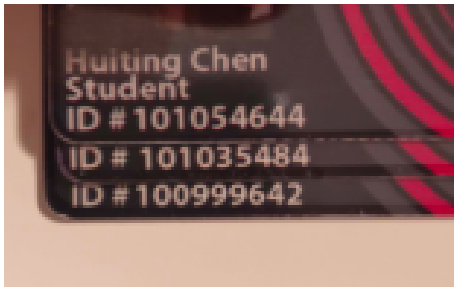

Fig. 14. first digit position inconsistent

Fig. 15. abrasion example

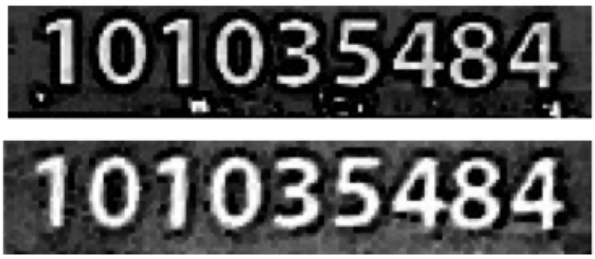

Fig. 16. resolution comparison after enhance and sharpen

5 Statement of Contributions

-Jiahe Geng: Primary role was to implement the algorithm to recognize the digits on the campus card and identify reference image. Also involved in presentation. Finally contributed to the writing of the report by discussing the background, proposed approaches, mostly for digits recognition.

-Yuhua Chen: Primary role was to implement the algorithm to find edges of the campus card and part of user interface, also involved in presentation and demo. Finally contributed to the writing of the report Introduction, Proposed approaches and Result section.

-Jiacheng Tang: Responsible for creating and designing the database and connecting it with the system. He also completed the insert, update and delete functions. And designed the user interface, completed the UI of the insert, update and delete parts.

6 GitHub Page

https://github.com/JiaheG/Campus-Card-Recognition

References

[1] Suzuki, S., & Be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics and Image Processing, 30(1), 32–46. https://doi.org/10.1016/0734-189X(85)90016-7

[2] OpenCV Dicumentation, https://opencv.org/

[3] Template Matching, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html

[4] Peng, C. (2015). Position-weighted template matching for measuring in-plane dynamics of microdevices (T). University of British Columbia. Retrieved from https://open.library.ubc.ca/collections/ubctheses/24/items/1.0220526

[5] Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[6] SURF https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[7] FLANN based Matcher https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[8] Credit card OCR with OpenCV and Python https://www.pyimagesearch.com/2017/07/17/credit-card-ocr-with-opencv-and-python/#download-the-code