A Better Laundry System for HKU Dorms

Atiab Bin Zakaria (3035718230)
Ho Joshua (3035689831)

Supervised by Dr. Jia Pan

1 February 2023
Outline

• Motivation
• Objectives
• Methodology
• Progress
• Difficulties
• Next Steps
• Conclusion
Motivation

• Doing laundry is an unnecessarily inefficient process at HKU dorms

• Residents have no convenient way of knowing the availability of machines without physically going to the laundry room.

• Contrast this with numerous other university residences equipped with smart laundry systems
Objectives

• Tracking time remaining on a machine
• Making the information easily accessible to users
• Create a system where users can reserve machines
Methodology: Hardware

• Use a Raspberry Pi paired with a camera module to collect raw data about time remaining
• Convenient to not disrupt existing infrastructure
  • Implement a solution based on augmentation instead
Methodology: Hardware

- Wi-Fi Smart Plugs
  - Turn the machine on for use when the user reserves it (for proof of concept)
  - Detect voltage
- Controlled via Tuya API
  - Tuya: IoT platform
Methodology: Core

• Use optical character recognition (OCR) technology on images of the machine displays captured by the camera module

• Technologies identified:
  • Tesseract OCR
  • ssocr
Methodology: Backend

• Control the camera module (picamera Python package)
• Feed the captured photos to the OCR engine (ssocr/Tesseract)
• Make the recognized text available using an HTTP API endpoint (FastAPI)
Methodology: Frontend

• Design an easy-to-use UI using the JavaScript framework Next.js (based on the React library) to present the information to the users in the form of a web application

• Connected to the backend application for up-to-date information via the HTTP API endpoint(s)

• Users can access the interface through personal devices such as smartphones and laptops/desktops
Methodology: Overall
Workflow

- User reservation:

  User reserves machine from the app
  
  API request is sent to the FastAPI server to update the database e.g., user ID, start and end times of the session
  
  App updates its UI
  
  The PI is notified on the particular user who reserved the machine
Workflow

• User identification:

  1. Users provide the QR code to the Pi Camera
  2. Upon successful identification and correct time, the Pi sends API request to the FastAPI server to update the database.
  3. The smart plug is turned on via Tuya API which allows the user to use the machine.
Workflow

• Using the machine:

1. User starts using the machine
2. Voltage is detected from the smart plug indicating start of machine use
3. OCR starts extracting time left in the washer and sends the info to the FastAPI server
4. Time is reflected in the user app
5. The absence of voltage is detected indicating end of machine use
6. User is notified when their clothes are done being washed
Wireframe: Login
Wireframe: Home
Wireframe: Booking Date
Wireframe: Booking Time
Wireframe: QR Code
Wireframe: Time Remaining
Progress

• Raspberry Pi and camera module have been acquired
• GitHub organization has been created for code sharing
Progress

• Successfully powered up the acquired hardware
  • Able to access Raspberry Pi remotely using SSH
  • Able to capture photos using the camera module
• Basic architecture preliminarily tested in isolation
  • Able to run develop code and run it on the Raspberry Pi smoothly
Smart Plug API

```
curl --request GET "https://openapi.tuya.cn/v1.0/iot-03/devices/6c3e8b0f824770c4dfc9a/status" --header "sign_method: HMAC-SHA256" --header "client_id: 7cwclyhqa97s5w5xefm8" --header "t: 1675157057.006" --header "mode: cors" --header "Content-Type: application/json" --header "sign: CA4F06B7464BB5C48B1D69FEA2E66EB1B97659B4BA7A75462681075F47F11B7" --header "access_token: 9a2e1bda3bbec03bf13bb7b00f3cc8b9"
```
Progress

- Tesseract struggles to recognize seven-segment display fonts
  - Better suited to textual contexts in serif and sans-serif fonts
- In any case, generating result is slow with Tesseract on the Raspberry Pi
Progress

- Identified ssocr as the alternative
  - ssocr is short for Seven Segment Optical Character Recognition
  - i.e., specifically designed for this task
- Initial results are promising
  - in terms of accuracy and speed
- Need to tune exact parameters for photos from the camera module
Progress (SSOCR)

• Uses simple and completely deterministic algorithm
• Specialized for recognizing monochrome seven segment images
• Not machine learning nor artificial intelligence
• Consequence: Image input have constraints and needs to be pre-processed
Progress (SSOCR)

• Two processes
  • Segmentation
  • Character Recognition
Progress (SSOCR)

• Segmentation
  • Separates each digit in the image
  • Searches for the first line with some foreground pixels (Used as left border of a digit)
  • Searches for the first line with only background pixels (Used as right border of a digit)
  • Done vertically as well
  • Repeat until all digit segments are found

Border of each digit
Progress (SSOCR)

- Character Recognition
  - Done to each segment found
  - Each digit can have up to 7 parts
  - Number of parts and their position determines the digit (with the help of a lookup table)

E.g. The number 8 has all seven parts

E.g. The number 7 has three parts
Progress (SSOCR)

• Two main problems
  • Noise
  • Skewness

• Cannot change image to black and white and feed to OCR directly
Progress (SSOCR)

• Noise
  • Some other spots have same color as the digits used
  • Non digit objects will be assumed to be digits
  • Affects OCR

After extracting all green colors
Progress (SSOCR)

- Skewness
  - SSOCR assumes input image to not be skewed
  - Undesired results occur e.g. skewed digits may recognize two digits as one
Progress (SSOCR)

• Skewness
  • Often seven segment display are naturally skewed
  • E.g. The slanted lower part of 7 affects its accuracy
Progress (SSOCR)

• Initial image pre-processing steps
  1. Color masking to extract only green color objects (the digits are green in HKU dorms)
  2. Make images black and white using thresholding
  3. Create bounding boxes for each object in the image (separate the digits from the noise i.e. solve noise problem)
  4. Rotate the bounding boxes if skewed (solve skewness problem)
  5. Feed each bounding box to SSOCR

• OpenCV is mainly used
Progress (SSOCR)

• Color masking
  • Extract a certain color from an image
  • Change to HSV color space
  • Only the green parts are shown

```python
import cv2
import numpy as np

img = cv2.imread("aaa.png")

# convert to hsv - more easy and accurate
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# mask of green (36,25,25) ~ (86, 255,255)
mask = cv2.inRange(hsv, (36, 25, 25), (86, 255,255))

# slice the green
imask = mask>0
green = np.zeros_like(img, np.uint8)
green[imask] = img[imask]
```
Progress (SSOCR)

• Making images black and white (thresholding)
  • Thresholding:
    • Turn pixels with greyscale value > 128 white else black
  • Change to greyscale values first
  • Use 128 as threshold value
    • The green color used has a greyscale value > 128

```python
image_color = cv2.imread("green.png", cv2.IMREAD_GRAYSCALE)
img_bw = cv2.threshold(image_color, 128, 255, cv2.THRESH_BINARY)[1]
```
Progress (SSOCR)

• Creating bound boxes
  • One small problem encountered
    • Some seven segment digits may have disconnected parts
    • Bounding box cannot contain the whole digit
Progress (SSOCR)

• Creating bound boxes
  • Solution is to dilate the image
    • Dilation: add more pixels to fully connect the parts of a digit

```python
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
image_color = cv2.dilate(img_bw,kernel,iterations=1)
```
Progress (SSOCR)

• Creating bound boxes

After dilation:  

Before dilation:
Progress (SSOCR)

• Creating bound boxes
  • Fix skewness of each bounding box before feeding to SSOCR
  • For each bounding box:
    • Get angle of rotation of the box (with opencv function)
    • Rotate original image, the skewed bounding box then crop
    • Finally feed to SSOCR to see if it’s a digit
    • Gather all digits to form the time remaining of machine
Progress (SSOCR)

```
for cnt in contours:
    rec = cv2.minAreaRect(cnt)
    # cropped = crop_rect(img, rec)
    plt.imshow(cropped[0])
    plt.show()
```

```
def crop_rect(img, rect):
    # get the parameter of the small rectangle
    center, size, angle = rect[0], rect[1], rect[2]
    center, size = tuple(map(int, center)), tuple(map(int, size))

    # get row and col num in img
    height, width = img.shape[0], img.shape[1]

    # calculate the rotation matrix
    M = cv2.getRotationMatrix2D(center, angle, 1)

    # rotate the original image
    img_rot = cv2.warpAffine(img, M, (width, height))

    # now rotated rectangle becomes vertical, and we crop it
    img_crop = cv2.getRectSubPix(img_rot, size, center)
    return img_crop, img_rot
```
Progress (SSOCR)

• Summary

Original  →  Color Masking  →  Black and White  →  Bounding Box (Without dilation)

Bounding box (With dilation)  →  Rotation + Crop

41
Progress (SSOCR)

• More testing to be done on more images
• Confident that these are the only 2 main problems
Next Steps

• Start developing applications for both the backend and frontend
• Look into an e-payment solution proof-of-concept if time allows
Conclusion

• HKU dorm residents are busy enough
• Inconvenient laundry does not have to add to their worries
• Ours is an inexpensive solution to reduce the overhead associated with laundry
  • While avoiding having to overhaul existing infrastructure