OCR for Captchas

DIP Final project - Group 21

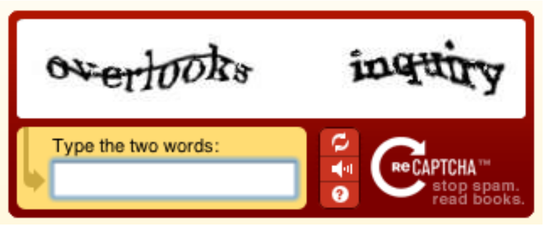
D01944015 Sebastian Agethen (蔡格昇)
A01922201 Jeroen Dhondt (唐杰)
R99222030 林昇慶

Content

- Introduction
- Motivation
- Optical Character Recognition
  - Structure & Framework
  - Features
  - Decision mechanism
- Demo
- Future Work

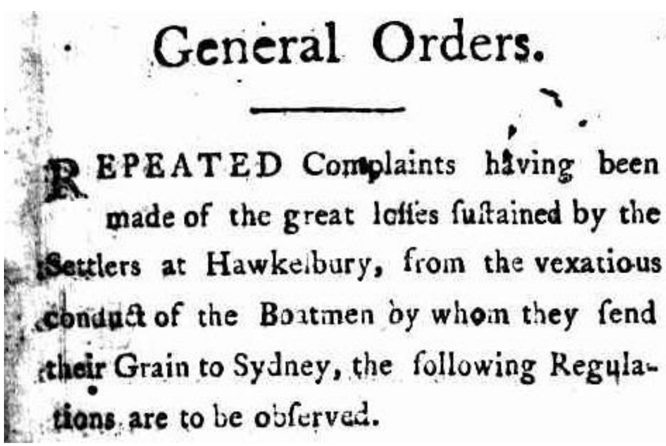
Intro: Captcha's

- What?
  - image containing distorted characters
  - difficult to recognize by machine
- Purpose?
  - confirm user is human
  - prevent automated bots to spam website etc.
- Examples:



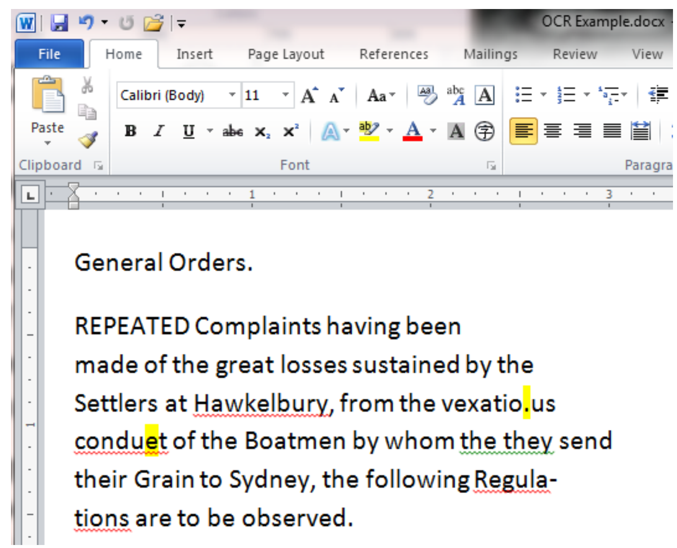
Optical Character Recognition (OCR): Intro

- <u>What?</u>
  - electronical conversion of images to text
  - recognize and distinguish different characters



General Orders.

REPEATED Complaints having been made of the great loſſes ſuſtained by the Settlers at Hawkeſbury, from the vexatious conduct of the Boatmen by whom they ſend their Grain to Sydney, the following Regula-tions are to be obſerved.



OCR: Intro(2)

- Applications?
  - digitize old paper documents
  - analyze pictures and extract information
  - ... solving captcha's?

Motivation

Final goal: solve Captcha's

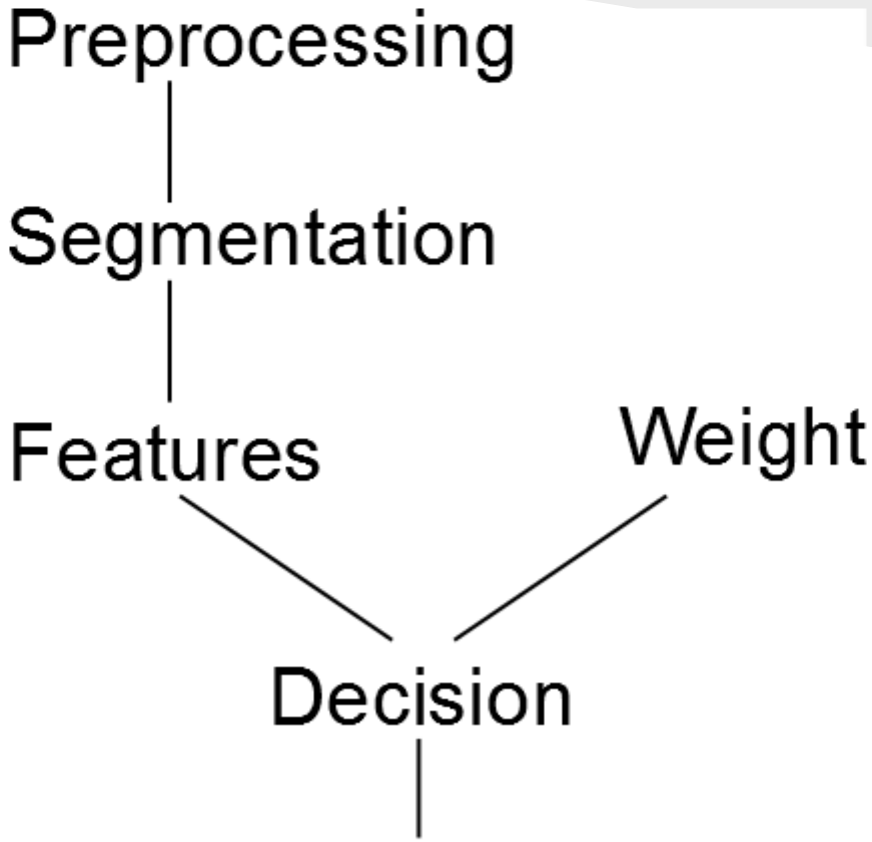
1. implement OCR application recognize text
2. process Captcha's and solve them
    - preprocess & remove distortions
    - use OCR to recognize characters

Optical Character Recognition (OCR)

Two phase approach:

- Training phase
  - Create 36 Bitmaps of Capital letters and numbers
  - Compute features for each bitmap
- Live phase
  - Isolate characters ("Segmentation")
  - Compute features
  - Decision clustering
- Assumption
  - Bitmap input data
  - Same font size (12pts)
  - Same font style (Arial)

OCR - Structure



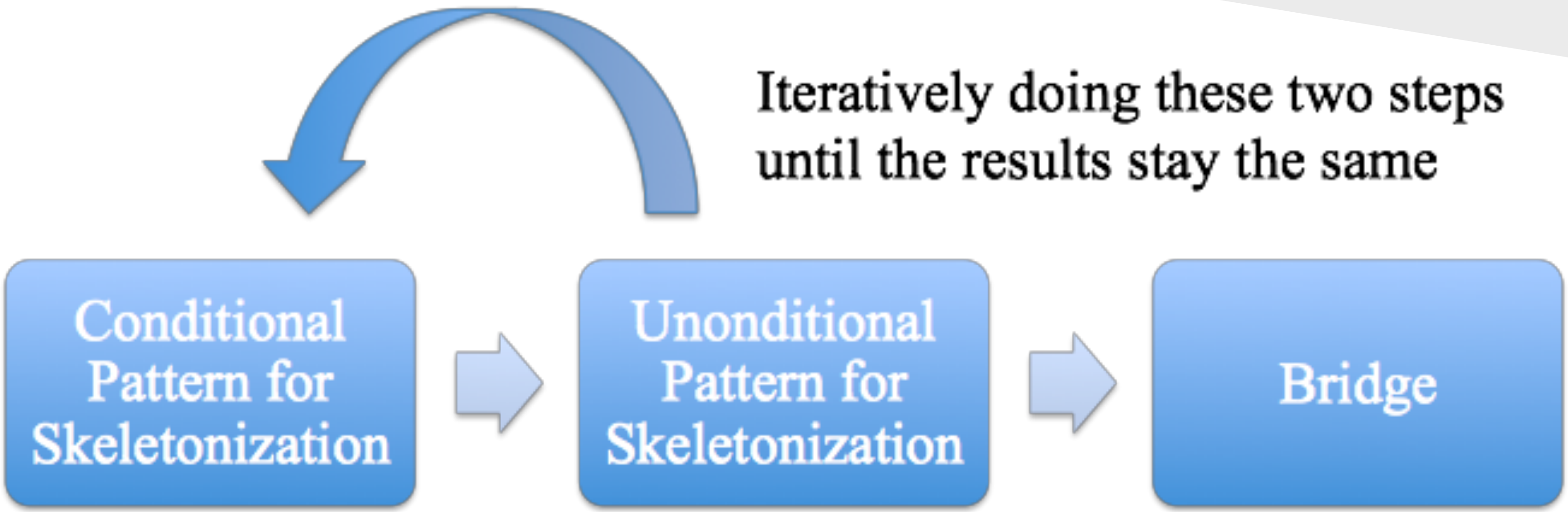
OCR - Preprocessing Techniques

1. Quantization (Black and White)
2. Possibly: Histogram equalization
3. Skeletonization
4. Boundary extraction
5. Convex Hull

Sentences/Characters Extraction

- Scan the whole input image from left to right
- Detect the gradient of the pixel
  - A strong gradient means that it is a boundary
- Problems
  - The quantisation threshold
  - The quality of the input data

Skeletonization



Iteratively doing these two steps until the results stay the same

Decide a pixel is a hit or miss
- Miss: stay the same
- Hit: a candidate to remove

Check the candidate pixel
- Miss: set to zero
- Hit: stay the same

Reconstruct the connectivity

Orientation

- The (m, n)th Spatial Moment

$$M(m,n) = \frac{1}{J^n K^m} \sum_{j=1}^{J} \sum_{k=1}^{K} (x_k)^m (y_j)^n F(j,k) \qquad \begin{cases} x_k = k - \frac{1}{2} \\ y_j = J + \frac{1}{2} - j \end{cases}$$

$$U(m,n) = \frac{1}{J^n K^m} \sum_{j=1}^{J} \sum_{k=1}^{K} (x_k - \frac{M(1,0)}{M(0,0)})^m (y_j - \frac{M(0,1)}{M(0,0)})^n F(j,k)$$

$$\lambda_M = MAX\{\lambda_1, \lambda_2\}$$

$$\lambda_1 = \frac{1}{2}[U(2,0) + U(0,2)] + \frac{1}{2}[U(2,0)^2 + U(0,2)^2 - 2U(2,0)U(0,2) + 4U(1,1)^2]^{1/2}$$

$$\lambda_2 = \frac{1}{2}[U(2,0) + U(0,2)] - \frac{1}{2}[U(2,0)^2 + U(0,2)^2 - 2U(2,0)U(0,2) + 4U(1,1)^2]^{1/2}$$

$$\Rightarrow \qquad \theta = \arctan\left\{ \frac{\lambda_M - U(0,2)}{U(1,1)} \right\}$$

Features

1. Area, diameter, etc
2. Bays and Lakes
3. Euler number
4. Line components, Circles
5. Shape Context (Belongie et al.)

Area, Weight Center, Diameter

- Area
  - Calculate the number of black pixel
- Weight Center

$$F_c = \frac{\sum F(i,j) x_{i,j}}{\sum x_{i,j}}$$

- Average distance from weight center
- Maximum distance from weight center

Bays, Lakes, Euler number

- Find #holes and #components!
- Holes: White connected components
  - That don't touch image borders
- Black connected components

- 2 white components, 1 black component
- E(8) = 1-2 = -1

Lines & Circles

- Find skeleton (or boundary) of letter
  - All lines are 1px strong
- A run of pixels in a dimension is a line

- Vertical lines (w/ minimum length): 3
- Horizontal lines: 6

Lines & Circles

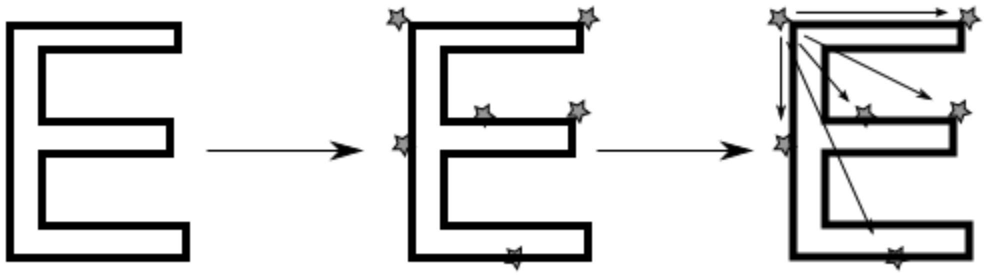
- Find skeleton (or boundary) of letter
  - All lines are 1px strong

- Circularity
  - Find a lake, check circularity coefficient
  - A: Area, P: Perimeter

$$C_0 = \frac{4\pi A_0}{(P_0)^2}$$

Shape Context

- Select n black samples (from contour)
  - Boundary extraction
- Compute log distance
  - Far pixels are more interesting in terms of shape!



Shape Context

- For each sample: Distance histogram
- Compare two histograms g, h of distance
  - Chi-Square distance

$$C_S = \frac{1}{2} \sum_{k=1}^{K} \frac{[g(k) - h(k)]^2}{g(k) + h(k)}$$

- But we need one value, not #sample values!
  - Match samples with bipartite matching
  - Apply some linear function (e.g., sum)

Decision mechanism

- Use one iteration of clustering
- For each bitmap in the training phase
  - Create a cluster centroid
- In live phase
  - Compute distances to each centroid
  - Choose closest centroid as result
- Choosing k-next-neighbors also possible
  - Post-processing

Demo!

Future Work

- Improve accuracy OCR:
  - find optimal weight coefficients
  - add additional properties
- Solve captcha's
  - process distorted image to suitable input OCR application

References

- "Shape Context: A new descriptor for shape matching and object recognition", Belongie et al., NIPS 2000
- "Breaking a Visual CAPTCHA", Greg Mori and Jitendra Malik, online source: http://www.cs.sfu.ca/~mori/research/gimpy