Recognizing Handwritten Source Code

Qiyu Zhi and Ronald Metoyer
University of Notre Dame
How about **programming**?
Programming on mobile devices is not unprecedented
Tillmann et al. predict that programming on mobile devices will be widely used in the future.
Buffer editor” App in iTunes store
Why handwriting?

Software developers who suffer from repetitive strain injuries (RSI) and related disabilities may find typing on a keyboard difficult. Instead, handwriting with a stylus may be a preferred input mechanism for some of these users.

Why handwriting?

Handwriting has also been shown to have cognitive benefits.

What is the key problem?

Handwriting Recognition

Is handwriting recognition good enough to use regularly on the iPad Pro?

We've come a long way from the Newton, but I'm still skeptical. Even with the Apple Pencil making it easier to get precise writing that looks semi-coherent, the app support needs to be there. Right now, it's not. There are two developers with handwriting recognition software on the App Store — Viet Tran and MyScript — but neither options are perfect.
Improvements in RWTH’s system for off-line handwriting recognition

Michael Kenndahl, Patrick Douchet, Herrman Ney
Human Language Technology and Pattern Recognition Group
Chair of Computer Science B
RWTH Aachen University, D-52056 Aachen, Germany
{kenndahl, douchet, ney}@cs.rwth-aachen.de

ABSTRACT
Off-line handwriting recognition remains challenging. The system we report in this paper obtained good results in the IAM-2008 challenge, the first international competition on off-line handwriting recognition. The system combines several approaches to re-construct the trajectory of the pen and to recognize the symbols presented thereon. It has been designed using state-of-the-art techniques from the domains of large vocabulary speech recognition, speech recognition, and image processing. We adopt successful techniques from these domains to build a new system for off-line handwriting recognition. The system is able to recognize handwritten text of various fonts, sizes, styles, orientations, and handwriting styles. It is able to recognize text written on paper, on overhead transparencies, and on electronic devices such as tablets.

1. INTRODUCTION

Handwritten text is an important medium for communication. It is used in many applications such as legal and business documents, personal notes, and educational materials. Furthermore, handwritten text is a key component of many digital systems, such as electronic health records, financial documents, and educational materials. The ability to recognize handwritten text is important for many applications, including text-to-speech synthesis, text mining, and information retrieval.

2. METHODS

2.1 Feature Extraction

The feature extraction process is critical for the success of a handwriting recognition system. In our system, we use a combination of optical flow and statistical approaches to extract features. Optical flow is a technique for estimating the motion of objects in a video sequence. It is particularly useful for tracking the movement of the pen across the page. Statistical approaches, such as hidden Markov models (HMMs) and support vector machines (SVMs), are used to model the relationship between the features and the symbols.

2.2 Symbol Recognition

The symbol recognition process is responsible for identifying the symbols written in the handwritten text. In our system, we use a combination of HMMs and SVMs to recognize the symbols. HMMs are used to model the temporal and spatial relationships between the symbols, while SVMs are used to classify the symbols based on their features.

2.3 Text Recognition

The text recognition process is responsible for recognizing the text written in the handwritten document. In our system, we use a combination of HMMs and SVMs to recognize the text. HMMs are used to model the temporal and spatial relationships between the symbols, while SVMs are used to classify the text based on the symbols.

3. RESULTS

The system we report in this paper obtained good results in the IAM-2008 challenge. The results were compared to those of other systems that participated in the challenge. Our system achieved a higher accuracy than most of the other systems in the challenge. The results were encouraging and showed that our system is effective in recognizing handwritten text.

4. CONCLUSION

In conclusion, we have presented a new system for off-line handwriting recognition. The system combines several successful techniques from the domains of large vocabulary speech recognition, speech recognition, and image processing. The system is able to recognize handwritten text of various fonts, sizes, styles, orientations, and handwriting styles. It is able to recognize text written on paper, on overhead transparencies, and on electronic devices such as tablets. The results of our system were promising and showed that our system is effective in recognizing handwritten text.

Cites


This work was supported by the European Commission under the 7th Framework Programme (FP7) under grant agreement no. 320159.

PDP - Pen Driven Programming

Jonathan Frye
The University of Edinburgh
School of Informatics
EH1 8LE

ABSTRACT
Pen-driven programming refers to the use of a pen to interact with a computer, allowing for a more natural and intuitive interaction with digital devices. This approach is particularly useful in scenarios where traditional keyboard and mouse input may be less efficient or practical, such as in portable devices or interactive environments. This paper describes the development of a pen-driven programming framework that enables users to write and edit code using a digital pen, providing a seamless integration of handwriting and programming. The system includes a handwriting recognition component that converts the pen-drawn code into machine-readable text, allowing for dynamic modifications and updates. The framework supports a range of programming languages, enabling users to experiment with different syntax and styles. The system has been evaluated through a series of user studies, demonstrating its effectiveness and potential for enhancing the programming experience. The results indicate that pen-driven programming offers a promising alternative to traditional input methods, offering increased productivity and a more natural interaction with digital devices. This approach has the potential to revolutionize the way we interact with technology, providing a more intuitive and engaging experience for users.

Keywords: pen-driven programming, handwriting recognition, interactive programming, natural user interface
Little research has paid attention to handwritten source code recognition.
We developed a handwritten source code recognition system by combining programming language knowledge with a handwriting recognition engine.
Data Collection

Characterizing Errors

Our Algorithm And Result

Discussion
Data Collection

Characterizing Errors

Our Algorithm And Result

Discussion
We decided to collect handwritten *Python* source code because of the **current popularity** of *Python* and its **projected growth rate**.

Which Language

Most Popular Coding Languages of 2015

- Python: 31.2%
- Java: 19.6%
- Javascript: 6.5%
- C: 6.1%
- PHP: 3.6%
- C#: 7.4%
- Ruby: 7.1%
- C++: 9.8%
What to write

We need to **find representative samples** of handwritten *Python* source code need to evaluate the performance of a handwriting recognition system.
What to write

Find 6 python projects from Github

Extract all functions after eliminating comments

Filter out functions that can’t fit in the screen

What to write

Find 6 python projects from Github

Extract all functions after eliminating comments

Filter out functions that can’t fit in the screen

What to write

Find 6 python projects from Github

Extract all functions after eliminating comments

Filter out functions that can’t fit in the screen

What to write

<table>
<thead>
<tr>
<th>Project</th>
<th>Lines</th>
<th>Functions</th>
<th>Eligible Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlphaGo</td>
<td>1,963</td>
<td>151</td>
<td>1</td>
</tr>
<tr>
<td>Bittorrent</td>
<td>7,164</td>
<td>570</td>
<td>39</td>
</tr>
<tr>
<td>Blender</td>
<td>265,684</td>
<td>12,774</td>
<td>1,126</td>
</tr>
<tr>
<td>Instagram</td>
<td>1,265</td>
<td>145</td>
<td>8</td>
</tr>
<tr>
<td>Requests</td>
<td>14,009</td>
<td>862</td>
<td>84</td>
</tr>
<tr>
<td>Webpy</td>
<td>10,199</td>
<td>1,029</td>
<td>66</td>
</tr>
</tbody>
</table>

Table 1: *Python* projects used for selecting code samples
3 + 1
def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if cookie.name == name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    if toReturn is not None:
                        raise CookieConflictError('There are , %r' % (name))
                    toReturn = cookie.value
    if toReturn:
        return toReturn
    raise KeyError(name, domain, path)

def take(seq, n):
    for i in range(n):
        yield next(seq)
    if not hasattr(seq, 'next'):
        seq = iter(seq)
    while True:
        x = list(take(seq, size))
        if x:
            yield x
        else:
            break

def calc_median_bbox_pivots(positions):
    median = None
    bbox = [None, None]
    n = 0
    for pos in positions:
        extend_bbox(bbox, pos)
    try:
        median += pos
    except:
        median = pos.copy()
    n += 1
    median = median / n
    bbox_center = (Vector(bbox[0]) + Vector(bbox[1])) * 0.5
    return median, bbox_center
def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if cookie.name == name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    if toReturn is not None:
                        raise CookieConflictError('There are , %r' % (name))
                    toReturn = cookie.value
                    return toReturn
    raise KeyError(name, domain, path)

def take(seq, n):
    for i in range(n):
        yield next(seq)
    if not hasattr(seq, 'next'):
        seq = iter(seq)
    while True:
        x = list(take(seq, size))
        if x:
            yield x
        else:
            break

def calc_median_bbox_pivots(positions):
    median = None
    bbox = [None, None]
    n = 0
    for pos in positions:
        extend_bbox(bbox, pos)
        try:
            median += pos
        except:
            median = pos.copy()
        n += 1
    median = median / n
    bbox_center = (Vector(bbox[0]) + Vector(bbox[1])) * 0.5
    return median, bbox_center

def asVertsLocation(verts2d, faces):
    t = clock()
    uv = []
    for f in faces:
        uvface = []
        for vi in f:
            uvface.extend(verts2d[vi])
        uv.append(uvface)
    print('uvs convert in', clock() - t)
    return uv

def read_lwostring(raw_name):
    i = raw_name.find(b'\0')
    name_len = i + 1
    if name_len % 2 == 1:
        name_len += 1
    if i > 0:
        name = raw_name[0:i].decode("utf-8", "ignore")
    else:
        name = ""
    return name, name_len

def _GetInfoRNA(bl_rna, cls, parent_id=""):
    if bl_rna is None:
        return None
    key = parent_id, bl_rna.identifier
    try:
        return cls.global_lookup[key]
    except KeyError:
        instance = cls.global_lookup[key] = cls(bl_rna)
        return instance

def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if cookie.name == name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    if toReturn is not None:
                        raise CookieConflictError('There are , %r' % (name))
                    toReturn = cookie.value
                    return toReturn
    raise KeyError(name, domain, path)

def take(seq, n):
    for i in range(n):
        yield next(seq)
    if not hasattr(seq, 'next'):
        seq = iter(seq)
    while True:
        x = list(take(seq, size))
        if x:
            yield x
        else:
            break

def calc_median_bbox_pivots(positions):
    median = None
    bbox = [None, None]
    n = 0
    for pos in positions:
        extend_bbox(bbox, pos)
        try:
            median += pos
        except:
            median = pos.copy()
        n += 1
    median = median / n
    bbox_center = (Vector(bbox[0]) + Vector(bbox[1])) * 0.5
    return median, bbox_center

def asVertsLocation(verts2d, faces):
    t = clock()
    uv = []
    for f in faces:
        uvface = []
        for vi in f:
            uvface.extend(verts2d[vi])
        uv.append(uvface)
    print('uvs convert in', clock() - t)
    return uv

def read_lwostring(raw_name):
    i = raw_name.find(b'\0')
    name_len = i + 1
    if name_len % 2 == 1:
        name_len += 1
    if i > 0:
        name = raw_name[0:i].decode("utf-8", "ignore")
    else:
        name = ""
    return name, name_len

def _GetInfoRNA(bl_rna, cls, parent_id=""):
    if bl_rna is None:
        return None
    key = parent_id, bl_rna.identifier
    try:
        return cls.global_lookup[key]
    except KeyError:
        instance = cls.global_lookup[key] = cls(bl_rna)
        return instance

3 + 1
def calc_median_bbox_pivots(positions):
    median = None
    bbox = [None, None]
    n = 0
    for pos in positions:
        extend_bbox(bbox, pos)
    try:
        median += pos
    except:
        median = pos.copy()
    n += 1
    median = median / n
    bbox_center = (Vector(bbox[0]) + Vector(bbox[1])) * 0.5
    return median, bbox_center

def take(seq, n):
    for i in range(n):
        yield next(seq)
    if not hasattr(seq, 'next'):
        seq = iter(seq)
    while True:
        x = list(take(seq, size))
        if x:
            yield x

def find_no_duplicates(self, name, domain=None, path=None):
    to_return = None
    for cookie in iter(self):
        if cookie.name == name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    if to_return is not None:
                        raise CookieConflictError('There are 1 or more ' + name)
                    to_return = cookie.value
                    if to_return:
                        return to_return
                        raise KeyError(name, domain, path)
Data Collection

15 participants
At least two semesters programming experience
8 had used a pen for touch screen devices
def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if cookie.name == name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    if toReturn is not None:
                        raise CookieConflictError('There are %r or %r (name) %r (value)')
                    toReturn = cookie.value
                    if toReturn:
                        return toReturn
                    raise KeyError(name, domain, path)
Procedure

Given 4 code samples

Handwrite code samples on iPad

```python
def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if cookie.name:
            if domain is None or cookie.domain == domain:
                if path is None or cookie.path == path:
                    raise CookieConflictError(f"There are 97 or 97 (name)\n"
                    toReturn = cookie.value
                    if toReturn:
                        return toReturn
                    raise KeyError(name, domain, path)
```
Data Collection Result

15 copies of **3 common** handwritten Python code samples

```python
def _find_no_duplicates(self, name, domain=None, path=None):
    toReturn = None
    for cookie in iter(self):
        if domain is None or cookie.domain == domain:
            if path is None or cookie.path == path:
                if not toReturn:
                    raise CookieConflictError("There are 2 or more " name)''
                toReturn = cookie.value
                if toReturn:
                    return toReturn
raise KeyError(name, domain, path)
```
Data Collection

Characterizing Errors

Our Algorithm And Result

Discussion
How well do existing state-of-the-art handwriting recognition systems perform on handwritten source code? In what ways do they fail?
Errors Categories

Word Errors

Symbol Errors

Space Errors

Symbol Errors

Data Collection

Characterizing Errors

Our Algorithm And Result

Discussion
“A programming language is governed by grammar rules, which stipulate the positions of keywords and symbols.”

“Programming languages are highly repetitive with predictable properties, especially for function names and variable names.”

Augmented MyScript Pipeline
Augmented MyScript Pipeline

MyScript CDK returns initial recognition result
Augmented MyScript Pipeline

Statement Classification:

Classify statement by first word

<table>
<thead>
<tr>
<th>Class</th>
<th>Frequency</th>
<th>Class</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>def</td>
<td>3</td>
<td>except</td>
<td>1</td>
</tr>
<tr>
<td>if</td>
<td>7</td>
<td>while</td>
<td>1</td>
</tr>
<tr>
<td>for</td>
<td>3</td>
<td>try</td>
<td>1</td>
</tr>
<tr>
<td>raise</td>
<td>2</td>
<td>break</td>
<td>1</td>
</tr>
<tr>
<td>return</td>
<td>2</td>
<td>else</td>
<td>1</td>
</tr>
<tr>
<td>yield</td>
<td>2</td>
<td>assignment</td>
<td>13</td>
</tr>
</tbody>
</table>
Augmented MyScript Pipeline

Statement Parsing:

According to proper grammar structure in this class (regular expression)
Augmented MyScript Pipeline

Token processing:

Processes space and variables (repetition)
Augmented MyScript Pipeline

Statement Concatenation:
concatenate words into statement (symbols)
Evaluation

Word Error Rate (WER)

Character Error Rate (CER)
Experiment Result

Word Error Rate (WER) 31.31%
Character Error Rate (CER) 9.24%
How good is the recognition rate? Is it acceptable?
Comparison to general handwriting recognition system

<table>
<thead>
<tr>
<th>System</th>
<th>WER(%)</th>
<th>CER(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Augmented MyScript</td>
<td>8.6</td>
<td>3.6</td>
</tr>
<tr>
<td>Kozielski et al. [9]</td>
<td>9.5</td>
<td>2.7</td>
</tr>
<tr>
<td>Keysers et al. [21]</td>
<td>10.4</td>
<td>4.3</td>
</tr>
<tr>
<td>Zamora et al. [45]</td>
<td>16.1</td>
<td>7.6</td>
</tr>
<tr>
<td>Poznanski et al. [34]</td>
<td>6.45</td>
<td>3.44</td>
</tr>
</tbody>
</table>

Recognition rate for code
Recognition rate for general English
Data Collection

Characterizing Errors

Our Algorithm And Result

Discussion
Remaining Errors

How do we fix it?

• Develop an interactive widget to highlight errors, then let user fix it.

• Train a language model to identify English word.
Generalizing to other languages

This approach can be generalized to other programming languages with strict grammar rules.
Expected use for our system

Handwriting speeds are commonly in the 15 to 25 wpm range.

In professional programming, most of the code that developers write involves reuse of existing example code and libraries.

We envision our system as being particularly useful in code editing and short code writing.
Future topics

Develop a handwritten source code recognition system from scratch

Explore the integration of handwriting recognition into source code IDEs

Combination of handwriting and speech input or handwriting and occasional keyboard input
Contribution summary

Collected handwritten python code samples from 15 participants.

Characterized recognition errors with general handwriting recognition system.

Developed a handwritten recognition system for source code.
Questions?

Collected dataset can be downloaded at:
http://www.purl.org/recognizinghandwrittencode/data.

The source code for augmented MyScript can be downloaded at:
http://www.purl.org/recognizinghandwrittencode/code.