Yo: A Video Analytical Editing Programming Language

Mengqing Wang, Munan Cheng, Tiezheng Li, Yufei Ou

{mw3061,mc4081,tl2693,yo2265}@columbia.edu

1 Introduction

Yo is a user-friendly programming language for movie production. We offer the fastest and most efficient non-linear video editing and analyzing. Users can produce videos from varieties of sources such as images or existing video clips and apply system- or user-defined functions to perform seamless video editing such as clip construction, duration adjustment, subtitle burning. Besides, Yo provides strong self-defined libraries for digital video analysis, such as sentimental analysis and pattern recognition etc. In this light, Yo’s objective is to facilitate analytical editing on videos and less human effort needs to be involved.

2 Frame, Clip, Layer

The concepts of frame, clip and layer are as illustrated in Figure ???. These concepts are recursively defined but can be simplified as follows. A frame is seen as one of a sequence still images which compose a clip. It can be constructed directly from an image stored on the hard disk or an extract from an existing clip. Once a clip is assembled from a series of frames at a certain frame rate (usually 24 frames per second), it can be exported as the final product, or to be layered with other clips to form a new clip.

Below we show the common operations on these elements.
### 3 Features

To reduce the learning curve for new users, Yo scripts borrows much grammar from Python and C++. The user code would be compiled into C++ code (to be compiled by a C++ compiler) and executed utilizing a collection of C++ libraries.
such as **libopenshot**\(^1\).

**Yo** is planned to support the following language features:

1. Automatic garbage collection and easy interpolation with existing C++ code and libraries.
2. Maximum code cleanliness: indent blocking, newline instead of colon between statements.
3. Anonymous function.
4. Functional syntax sugar: with built-in functions such as `map`, `filter`, `lapply`, users can define powerful inline expressions.
5. Object-oriented programming.

### 4 Use Case

Now we show how **Yo** facilitates movie editing with a couple of examples.

#### 4.1 Storyboard

The first example involves concatenating a collection of images into a clip, joining it with a series of existing clips, and showing a subtitle on the most front layer.

---

**Example 1- Arrange clips and add subtitles**

```plaintext
# Read all images in directory "wd"
frames = [readFrame(f) for f in wd if f.endswith(".png")]

# Turn each Frame into a Clip of one second, and concatenate
# them into a new Clip called "clip_f"
clip_f = lapply(+, Clip(), [Clip(fm, dur=1.0) for fm in frames])

# Read all videos in directory "wd"
clips = [readVideo(f) for f in wd if f.endswith(".avi")]

# Join a part of clips[0] (4.5s to 12.5s) and clips[1]
# (starting from 3.5s till end) to "clip_f"
clip_v = clip_f + clips[0](4.5 : 12.5) + clips[1](3.5 :)
```

---

1. [https://launchpad.net/libopenshot](https://launchpad.net/libopenshot)
4.2 Effects

Next we apply quick color corrections to a part of the clip.

Example 2 - Add effects to frames

```python
# Define a function that recolors all pixels RGB(>140,?,<50)
def changeColor(fm):
    for row in fm.pixels:
        for p in row:
            if p.r > 140 and p.b < 50:
                p.r = 200
                p.b = 20
    return fm

# toClip is a built-in function that turns frames into a clip
# apply "changeColor" to part of clip
newClip = clip(5.6) + toClip(map(lambda x: changeColor(x), clip(5.6:15.6).frames())) + clip(15.6:)
```

4.3 Analysis

Here we show how Yo performs analytical editing. The below expression performs the following:

(a) Filter the clips and only keep those shorter than 10 seconds;
(b) Scale the clips to 720x360
(c) Trim off the first and the last second in each clip;
(d) Concatenate these clips into a video.

Example 3 - Calculate based on videos

```python
lapply(+, Clip(),
    [c(1.0:-1.0).scale(720) for c in clips if c.duration < 10.0]
)
```
The analytical editing becomes quite handy when we want to extract clip pieces of some interesting features from a long, everlasting (think of a security camera) video. For example, the following statement extract all clips with Yo’s appearance:

```r
Example 4- Identify Yo’s face :

```
```r
face = readFrame("yo_face.png")
yo_appearance = lapply(+, Clip(),

   [fm for fm in clips.frames() if imageMatch(fm, face) > 0.95])
```

## 5 Syntax

### 5.1 Types

Like a scripting language, users do not need explicit type declaration for variables. The equal sign `=` is used to assign values to variables. Types of variables are inferred and could be overwritten.

**Basic data types:**

1. `int` : signed integers
2. `double` : floating point real values
3. `bool` : boolean
4. `string` : a contiguous set of characters

**Composite data types:**

1. `array` : holds a sequence of elements of the same type
2. `tuple` : holds a sequence of elements, immutable
3. `struct` : a user-defined prototype for an object that defines a set of attributes including variables and methods

```r
Example 5- TypeExample

```
```r
answer = 42  # int
dertime = 7.5  # double
subtitle = "Yo,world"  # string
criteria = (a > 2)  # boolean
clips = []  # array
color = (255,136,23)  # tuple
clip = Clip()  # struct
```
5.2 Operators

Yo provides with operators as shown in Table 5.

5.3 Control Flow

Yo supports basic control flow. To help understand the code, the Yo code snippet is followed by C++ code that achieves the same effect.

Example 6 – Yo Control Flow Example

```java
# conditional statement
if clip.time > 10
  log(clip.time)

# cascading for-loop
for i <- 1 to 10, j <- 1 to 10, i + j == 10
  log("%d+%d=%d
", i, j, i + j)

s = 0
for i <- 10 downto 1 by -1, i != 2, x <- a[i]
  s += x

# suffix if/while/for
log("Yo world") if length > 100
fun1() while a > b
a[i] = 0 for i <- 1 to 10, i % 2 == 0
```

Example 7 – C++ Control Flow Example

```c++
int main() {
  if (clip.time > 10){
    printf("%d",clip.time);
  }
  for (auto i = 1; i <= 10; ++i) {
    for (auto j = 1; j <= 10; ++j) {
      if (i + j == 10) {
        printf("%d+%d=%d\n", i, j, i + j);
      }
    }
  }
  auto sum = 0;
  for (auto i = 10; i >= 1; i = i + -1) {
    printf("%d+%d=%d\n", i, j, i + j);
  }
}
```
if (i != 2) {
    for (auto x : a[i]) {
        sum += x;
    }
}
if (length > 100) {
    printf("Yo\nworld");
}
while (a > b) {
    fun1();
}
for (auto i = 1; i <= 10; ++i) {
    if (i % 2 == 0) {
        a[i] = 0;
    }
}

5.4 Functions

Yo functions are defined starting with keyword func. Lambda functions are also supported.

Example 8- Function Example

# function declaration
func longerTime (a, b) {
    if a.duration > b.duration {
        return a.duration
    } else {
        return b.duration
    }
}

# function calls
longerTime(clip1,clip2)

# lambda functions
pixels = map(lambda x -> x.r=200 && x.b=20, selected_pixels)
5.5 IO

Yo load video, audio and image files on the drive, render the timeline and output edited video. To debug and log, a log function that dumps standard output stream is provided.

<table>
<thead>
<tr>
<th>Example 9- IO Example</th>
</tr>
</thead>
<tbody>
<tr>
<td># read all images in a directory &quot;wd&quot;</td>
</tr>
<tr>
<td>frames = [readFrame(f) for f in wd if f.endswith(&quot;.png&quot;)]</td>
</tr>
<tr>
<td># read all videos in directory &quot;wd&quot;</td>
</tr>
<tr>
<td>clips = [readVideo(f) for f in wd if f.endswith(&quot;.avi&quot;)]</td>
</tr>
<tr>
<td># log file name to stdout</td>
</tr>
<tr>
<td>log(clips[0].filename)</td>
</tr>
<tr>
<td># output final movie</td>
</tr>
<tr>
<td>saveClip(&quot;myYo.webm&quot;,final_cut)</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td>#{}</td>
</tr>
<tr>
<td>#}</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>*</td>
</tr>
<tr>
<td>/</td>
</tr>
<tr>
<td>%</td>
</tr>
<tr>
<td>&amp;</td>
</tr>
<tr>
<td>^</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>&lt;</td>
</tr>
<tr>
<td>&lt;=</td>
</tr>
<tr>
<td>==</td>
</tr>
<tr>
<td>&gt;=</td>
</tr>
<tr>
<td>&gt;</td>
</tr>
<tr>
<td>=</td>
</tr>
<tr>
<td>&lt;-</td>
</tr>
<tr>
<td>-&gt;</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>~</td>
</tr>
<tr>
<td>&quot;</td>
</tr>
<tr>
<td>[]</td>
</tr>
<tr>
<td>(</td>
</tr>
<tr>
<td>,</td>
</tr>
</tbody>
</table>

Table 2 - Operators and Notations in Yo.