YAnimoji: Yet Another Animoji Algorithm

CHEN Siyu

March 26, 2018

1 Preface

For math editing, the article is written in LaTeX. Unfortunately, my current TeXenvironment does not support Chinese inputs. As a result, sadly, the whole article will have to proceed in a foreign language which sources back to some ignominious pirates, predators, and colonists.

2 Utilized Information

2.1 Key Features

This subsection lists the name of the valid facial key features. The key features used are values that determines the expression of the avatar.

Valid features are:

- $EyeBink_L$
- $EyeBink_R$
- $EyeSquint_L$
- $EyeSquint_R$
- $EyeDown_L$
- $EyeDown_R$
- $EyeIn_L$
- $EyeIn_R$
- $EyeOpen_L$
- $EyeOpen_R$
- $EyeOut_L$
- $EyeOut_R$

- $EyeUp_L$
- $EyeUp_R$
- $BrowsD_L$
- $BrowsD_R$
- $BrowsU_C$
- $BrowsU_L$
- $BrowsU_R$
- $JawOpen$
- $LipsTogether$
- $JawLeft$
- $JawRight$
- $JawFwd$
- $LipsUpperUp_L$
- $LipsUpperUp_R$
- $LipsLowerDown_L$
- $LipsLowerDown_R$
- $LipsUpperClose$
- $LipsLowerClose$
- $MouthSmile_L$
- $MouthSmile_R$
- $MouthDimple_L$
- $MouthDimple_R$
- $LipsStretch_L$
- $LipsStretch_R$
- $MouthFrown_L$
- $MouthFrown_R$
- $MouthPress_L$
- $MouthPress_R$
- $LipsPucker$
- $LipsFunnel$
- $MouthLeft$
- $MouthRight$
- $ChinLowerRaise$
- $ChinUpperRaise$
- $Sneer_L$
- $Sneer_R$
- $Puff$
- $CheekSquint_L$
- $CheekSquint_R$

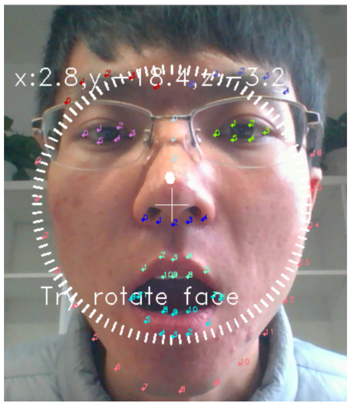

Figure 1: face landmarks

2.2 Landmarks

The figure shows the landmarks detected by the shape predictor.

The predictor used is based on ensemble of regression trees, which is sometimes referred as ERT.

The ensemble goes in two dimensions: Forests are used to predict a single point of the landmarks; Those forests aggregate together to estimate all the landmarks.

2.3 Calibrations

The first 1 second after the camera is opened is used to execute calibration. The calibration process save the status of a straight-frontal, neutral and emotion-less face. This status will be used later for comparisons in order to determine relative facial performances.

3 Computations

3.1 Relativity Function

$$rel(x, y, ignore, max, amplifier) \in [0, 1] \tag{1}$$

The relativity function measures the relative magnitude of x against y. $rel(x)$ becomes closer to 1 when x is greater. However, for de-noising purposes, the initial increment of x is ignored by $ignoreRatio$. Also, when the value of x exceeds max, the output will hold to 1.

3.2 Smooth Function

$$smooth(x_i) = \sum_{j=1}^{m} p(x_{i-j}) x_{i-j}, \tag{2}$$

in which $\sum p(x) = 1$.

The smooth function is used to suppress the noise brought by unstable regressions.

3.3 Initial Values

Since many face performances are relative by individual, there are a few values need to be saved as an referent. When you see $self.someValue$ in the code, that would just refer to the initial value of $someValue$ saved when calibrating faces.

3.4 Mapping Formulas

$$\begin{aligned}
leftchin &= named\_landmark[chin][1], & (3)\\
rightchin &= named\_landmark[chin][15], & (4)\\
jaw &= named\_landmark[chin][8], & (5)\\
forehead &= named\_landmark[nose\_bridge][0], & (6)\\
nosetip &= named\_landmark[nose\_bridge][3], & (7)\\
uplip &= named\_landmark[top\_lip][9], & (8)\\
downlip &= named\_landmark[bottom\_lip][9], & (9)\\
nose\_root &= named\_landmarks[nose\_bridge][0], & (10)\\
nose\_tip\_center &= named\_landmarks[nose_tip][2], & (11)\\
upper\_lip\_center &= nl['top\_lip'][9], & (12)\\
lower\_lip\_center &= nl['bottom\_lip'][9]. & (13)
\end{aligned}$$

3.5 Rotations

Firstly, the x,y,z rotation of head is given by:

$$\begin{aligned}
verticalRotation &= \frac{\|nosetip, forehead\|}{\|jaw - forehead\| - \|uplip - downlip\|}, & (14)\\
frontRotation &= \arctan(-\vec{rightchin} - \vec{leftchin}), & (15)\\
horizontalRotation &= \frac{nosetip[0] - faceHorizontalCenter}{rightchin[0] - leftchin[0]}. & (16)
\end{aligned}$$

Notably, the vertical rotation is based on the relative length of nose against the length of the entire face. Because of the diversity of nose shapes, the vertical rotation appears to be an relative value. This value is only comparable with the vertical rotation of the straight frontal face. Hence, that referential value should be, and is, saved for further comparison.

The values of xyz rotation are scaled into $[0, 1]$. Then the value can be mapped to a range in degrees (e.g. 60)

$left_{canthus}$, and $right_{canthus}$ respectively, is the intermediate position between the left-most corner of the left eye, and the nearest face boundary.

The purpose of doing such an interpolation is to reduce the shift of eye position brought by face horizontal rotation.

$$canthus_{distance} = \|left_{canthus} - right_{canthus}\|. \tag{17}$$

$$currentMouthCanthusRatio = \frac{mouthWidth}{canthusDistance} \tag{18}$$

currentMouthCanthusRatio is used to monitor the mouth movement according to the distance of eye-canthi.

3.6 Smiling And Frowning

$$\begin{aligned}
&current\_mouth\_left\_canthus\_ratio = && (19)\\
&\quad smooth(mouth\_left\_width/(canthus\_left\_to\_nose + 1e - 7)), && (20)\\
&mouthSmile = rel(current\_mouth\_left\_canthus\_ratio, && (21)\\
&\quad initalCurrentMouthLeftCanthusRatio) && (22)
\end{aligned}$$

in which, $initalCurrentMouthLeftCanthusRatio$ is the value of $CurrentMouthLeftCanthusRatio$ saved when calibrating/estimating the referential face geometry.

The distance between two canthi is used as an reference to measure the movement of mouth.

Mouth frown and mouth smile are mutual exclusive. Hence, the programme will check if smiling is detected. If not, the mouth frown detection will be set to valid.

$$\begin{aligned}
cos_{left} &= \frac{VleftMouthCorner \cdot v_{mouth}}{lenVmouth * lenVleftMouthCorner}, && (23)\\
sin_{left} &= \sqrt{1 - cos^2_{left}}. && (24)
\end{aligned}$$

The vector of mouth corner (V mouth corner) points to the middle point of mouth from the corner of mouth. When the vector points upward, the mouth corner is believed to be dropping, which helps capture mouth frownings. Here we measure the angle of mouth corners dropping from the horizontal line. The vertical droppings of mouth corner is calculated. Together with mouth length, the cos value of the mouth corner is calculated.

3.7 Lips Funneling

Lips funneling is detected also by inspecting the relative magnitude of variable $CurrentMouthLeftCanthusRatio$ against the initial value of it.

Jaw opening is estimated by measuring the distance from the center of upper lip to that of the lower lip.

$$mouth\_absolute\_percent = \frac{mouthHeight}{faceLength}. \tag{25}$$

The referent used here is the length of the entire face. The $mouth\_absolute\_percent$ is then scaled into the range of $[0, 1]$ and used to represent the degree of funneling.

3.8 Eyebrows

Eyebrow movement is measured by calculating the distance between the averaged gravity-center of all eyebrow key-points, and the center-of-gravity of all the lower-eye-lid key-points.

The averaging operations is applied mainly for stabilization.

Then, the relativity of eyebrow-height against the length of nose is computed. Scaling is performed, also, to obtain valid eyebrow performance measurements.

3.9 Facial Performance Binding

It is observed that there are some categories of facial performance which are not independent nor orthogonal. Some of the performances can even be linearly related. Hence, we picked some similar categories and bind them together to enrich the variety.

Lips pucker is bound to half of the Lips-Funnel value. $BrowsU_C$ is bound with $BrowsU_L$ and $BrowsU_R$ as shown below:

$$BrowsU_C = 0.8\frac{BrowsU_L + BrowsU_R}{2}. \tag{26}$$

Eye-open is bound to Eye-up action exactly.

4 Parameter Determining

Mathematically, our model solves regression problem with slack loss. Formally, we denote our model as a mapping function f with parameter θ and input x.

$$\begin{aligned} f(x|\theta) &= y, & (27) \\ {}^*\theta &= \arg\min |f(x) - y'|, & (28) \end{aligned}$$

in which, y represents output value and y' represents the ground-truth value.

To get optimal θ, we only need to ensure the mapping function f has the same monotonicity as ground-truth y with respect to input variable x.

Since the monotonicity is carefully designed in f, what we need to do is only to keep the direction, or sign, of the function correct. We can achieve that using the following method:

$$\begin{aligned} y'_i &= f(x_i|\theta) + \delta_i & (29) \\ {}^*\theta &= \arg\min \sum_{i=1}^{k} \delta_i, & (30) \end{aligned}$$

where we have k samples $(x_i, y'_i), i = 1, 2, ..., k$ roughly covering the domain as well as range of f.

After this step, we would have the parameter θ that can generate inference y with low KL-divergency to y'.

However, since some of the computations are not biased, a global bias shall be learned to complete the model.

Hence, another control-set is needed to perform more precise restrictions around zero boundary. This will make f more stable in respect of activation efficiency.

$$\begin{aligned} y'_i &= 0 + \gamma_i, i = 1, 2, ..., k \quad , & (31) \\ {}^*C &= \arg\min \sum_{i=1}^{k} f(x_i|\theta) + C. & (32) \end{aligned}$$

in which, γ_i is a slack value constraining the value of the control-samples chosen. γ is recommended to be set smaller than $1e-1$. C is the residual offset to be learned that will later be applied to f.

Now the trained model can be represented as:

$$g(x) = f(x|{}^*\theta) +^* C. \quad (33)$$