Notations

We denote :

- Pressure : $\nu \in [0,1]$
- time : $t \geq 0$
- point : defined by position x and y (or $p = (x,y) \in \mathbb{R}^2$)
- Raw inputs are denoted by a tuple $i[k] = (\nu[k], t[k], x[k], y[k])$ with $k \in \mathbb{N}$.

An *input stream* is a sequence of raw inputs $i = (\nu, t, x, y)$

- with time $t[k] \nearrow\nearrow$ strictly increasing
- starts with a `kDown` event
- contains `KMove` for $k \geq 1$
- ends either with a `kMove` or a `kUp` . If it is a `kUp` we say the input stream is *complete*

We addition define

- v_x, v_y, a_x, a_y as the velocity and acceleration.

With the vector shorthand $v = \left(v_x, v_y\right) \in \mathbb{R}^2$ and $a = \left(a_x, a_y\right) \in \mathbb{R}^2$

Wobble smoothing

To reduce high frequency noise.

Algorithm 1: Wobble smoothing

Input : $\{(x[k], y[k], t[k]) \in \mathbb{R}^2 \times \mathbb{R}_+, 0 \leq k \leq n\}$, $\Delta T > 0$ (from `wobble_smoother_timeout`), v_{min} (from `wobble_smoother_speed_floor`) and v_{max} (from `wobble_smoother_speed_ceiling`)

Compute a weighted moving average of the positions $\overline{p}[j] = (\overline{x}[j], \overline{y}[j])$

$$\forall j \in [\![0, n]\!], \overline{p}[j] = \begin{cases} \dfrac{\displaystyle\sum_{k=1}^{n} p[k](t[k] - t[k-1])\mathbb{1}_{\left[t[j]-\Delta T, t[j]\right]}(t[k])}{\displaystyle\sum_{k=1}^{n} \mathbb{1}_{\left[t[j]-\Delta T, t[j]\right]}(t[k])} & \text{if the numerator} \neq 0 \\ p[j] & \text{otherwise} \end{cases}$$

Calculate a moving average velocity $\overline{v}[j]$

$$\forall j \in [\![0, n]\!], \overline{v}[j] = \begin{cases} 0 & j = 0 \\ \dfrac{\displaystyle\sum_{k=1}^{n} \|p[k] - p[k-1]\| \mathbb{1}_{\left[t[j]-\Delta T, t[j]\right]}(t[k])}{\displaystyle\sum_{k=1}^{n} (t[k] - t[k-1])\mathbb{1}_{\left[t[j]-\Delta T, t[j]\right]}(t[k])} & \text{otherwise} \end{cases}$$

Interpolate between the average position and the raw ones based on the average speed

$$\forall j \in [\![0, n]\!], p'[j] = \min\left(\frac{\overline{v}[j] - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}} \mathbb{1}_{[v_{\text{min}}, \infty[}(\overline{v}[j]), 1\right)\overline{p}[j]$$
$$+ \left(1 - \min\left(\frac{\overline{v}[j] - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}} \mathbb{1}_{[v_{\text{min}}, \infty[}(\overline{v}[j])\right)\right)p[j]$$

where $p'[j] = (x'[j], y'[j])$

Output: $\{(x'[k], y'[k]) \in \mathbb{R}^2, 0 \leq k \leq n\}$ the filtered positions

---

Hence for low local speeds, the smoothing is maximum (we take exactly the average position over the time ΔT) and for high speed there is no smoothing. We also note that the first position is thus never filtered.

Resampling

Algorithm 2: Upsamling

Input : $\{v[k] = (x[k], y[k], t[k]), 0 \leq k \leq n\}$ and a target rate `sampling_min_output_rate`

Interpolate time and position between each k by linearly adding interpolated values This is done by adding linearly interpolated values so that the output stream is

$$\left\{ v[0], \underbrace{u_0[1], ..., u_0[n_0 - 1]}_{\text{interpolated}}, v[1], \underbrace{u_1[1], ..., u_1[n_1 - 1]}_{\text{interpolated}}, v[2], ... \right\}$$

Each n_i is the minimum integer such that

$$\frac{t[i] - t[i-1]}{n_i} < \Delta_{\text{target}}$$

$$\Leftrightarrow n_i = \left\lceil \frac{t[i+1] - t[i]}{\Delta_{\text{target}}} \right\rceil$$

and the linear interpolation means

$$u_j[k] = \left(1 - \frac{k}{n_j}\right) v[j] + \frac{k}{n_i} v[j+1]$$

Output : $\{(x'[k'], y'[k'], t'[k']), 0 \leq k; \leq n'\}$ the upsampled position and times. This verifies

$$\forall k', t'[k'] - t'[k'-1] < \Delta_{\text{target}} = \frac{1}{\texttt{sampling\_min\_output\_rate}}$$

Remark : As this is a streaming algorithm, we only calculate this interpolation with respect to the latest stroke position.

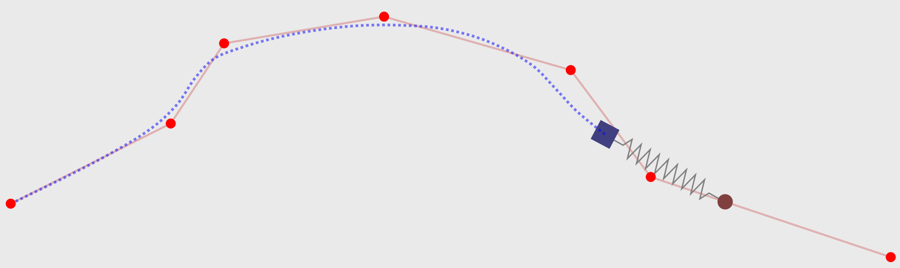
Position modeling

The raw input is processed as follow

raw input $\to$ wobble smoother $\to$ upsampled $\to$ position modeling

The position of the pen is modeled as a weight connected by a spring to an anchor.

The anchor moves along the *resampled dewobbled inputs*, pulling the weight along with it accross a surface, with some amount of friction. Euler integration is used to solve for hte position of the pen.



The physical model that is used to model the stroke is the following

$$\frac{\mathrm{d}^2 s}{\mathrm{d}t^2} = \frac{\Phi(t) - s(t)}{k_{\text{spring}}} - k_{\text{drag}} \frac{\mathrm{d}s}{\mathrm{d}t}$$

where

- t is time
- $s(t)$ is the position of the pen
- $\Phi(t)$ is the position of the anchor
- k_{spring} and k_{drag} are constants that sets how the spring and drag occurs

k_{spring} is given by `position_modeler_spring_mass_constant` and k_{drag} by `position_modeler_drag_constant`

We will thus have as input the *upsampled dewobbled inputs* taking the role of discretized $\Phi(t)$ and $s(t)$ will be the output

Modeling a stroke.

- Input : input stream $\{(p[k], t[k]), 0 \leq k \leq n\}$
- Output : smoothed stream $\left\{\left(p_f[k], v_f[k], a_f[k]\right), 0 \leq k \leq n\right\}$

We define $\Phi[k] = p[k]$. An euler scheme integration scheme is used with the initial conditions being $v[0] = 0$ and $p_f[0] = p[0]$ (same initial conditions)

Update rule is simply

$$a_f[j] = \frac{p[j] - p_f[j-1]}{k_{\text{spring}}} - k_{\text{drag}} v_f[j-1]$$

$$v_f[j] = v_f[j-1] + (t[j] - t[j-1]) a_f[j]$$

$$p_f[j] = p_f[j-1] + (t[j] - t[j-1]) v_f[j]$$

The position $s[j]$ is the main thing to export but we can also export speed and acceleration if needed. We denote

$$q[j] = \left(p_f[j], v_f[j], a_f[j], t[j]\right)$$

and this will be our output with $0 \leq j \leq n$

Stroke end

The position modeling algorithm will lag behind the raw input by some distance. This algorithm iterates the previous dynamical system a few additional time using the raw input position as the anchor to allow a `catch up` of the stroke (though this prediction is only given by `predict`, so is not part of the `results` and becomes obsolete on the next input).

Algorithm 3: Stroke end

Input:

- Final anchor position $p[\text{end}] = (x[\text{end}], y[\text{end}])$ ▷ From the original input stream
- final tip state $q_f[\text{end}] = \big(p_f[\text{end}] = \big(x_f[\text{end}], y_f[\text{end}]\big), v_f[\text{end}], a_f[\text{end}]\big)$ ▷ returned from the physical modeling from the last section, $\cdot_f$ signifies that we are looking at the filtered output
- $K_{\max}$ max number of iterations ▷ `sampling_end_of_stroke_max_iterations`
- Δ_{target} the target time delay between stroke ▷ `1/sampling_min_output_rate`
- d_{stop} stopping distance ▷ `sampling_end_of_stroke_stopping_distance`
- k_{spring} and k_{drag} the modeling coefficients

initialize the vector q_o with $q_0[0] = (\underbrace{p_f[\text{end}]}_{p_o[0]}, \underbrace{v_f[\text{end}]}_{v_o[0]}, \underbrace{a_f[\text{end}]}_{a_0[0]})$

initialize $\Delta t = \Delta_{\text{target}}$

for $1 \leq k \leq K_{\max}$

  calculate the next candidate

$$a_c = \frac{p[\text{end}] - p_o[\text{end}]}{k_{\text{spring}}} - k_{\text{drag}} v_0[\text{end}]$$

$$v_c = v_o[0] + \Delta t a_c$$

$$p_c = p_o[0] + \Delta v_c$$

  if $\|p_c - p[\text{end}]\| < d_{\text{stop}}$ ▷ further iterations won't be able to catch up and won't move closer to the anchor, we stop here

    return q_0

  endif

  if $\langle p_c - p_o[\text{end}], p[\text{end}] - p_o[\text{end}]\rangle < \|p_c - p_o[\text{end}]\|$ ▷ we've overshot the anchor, we retry with a smaller step

    $\Delta t \leftarrow \frac{\Delta t}{2}$

    continue ▷ this candidate will be discarded, try again with a smaller time step instead

  else

    $q_0[\text{end} + 1] = (p_c, v_c, q_c)$ ▷ We append the result to the end of the q_0 vector

  endif

  if $\|p_c - p[\text{end}]\| < d_{\text{stop}}$

    return ▷ We are within tolerance of the anchor, we stop iterating

  endif

  Output : $\{q_o[k] = (s_o[k], v_o[k], a_o[k]), 0 \leq k \leq n (\leq K_{\max} - 1)\}$

Stylus state modeler

Up till now we have only used the raw input stream to create a new smoothed stream of positions, leaving behind the pressure attribute. This is what's done here, to model the state of the stylus for these new position based on the pressure data of the raw input strokes.

Algorithm 4: Stylus state modeler

Input :

- input stream with pressure information $\{(p[k] = (x[k], y[k]), \nu[k]), 0 \leq k \leq n\}$
- query position $q = (x, y)$
- search window n_{search} ▷ From `stylus_state_modeler_max_input_samples`,

initialize $d = \infty$, index $=$ None, interp $=$ None
for $i = n - n_{\text{search}}$ to $n - 1$ **do**
  Find q_i the position that's closest to q on the segment $[p[i], p[i+1]]$ and denote $r \in [0, 1]$ the value such that $q_i = (1 - r)p[i] + rp[i + 1]$
  if $\|q - q_i\| < d$
    $d \leftarrow \|q - q_i\| < d$
    index $= i$
    interp $= r$
    endif
  endfor
  calculate

$$\nu = (1 - r)\nu[\text{index}] + r\nu[\text{index} + 1]$$

Output : interpolated pressure ν