Active Contour

September 2021

1 Objective

Given an image $\boldsymbol{P}$ and two key points (x_0, y_0) and (x_n, y_n), we have a loss function:

$$L(x_1, y_1, \ldots, x_{n-1}, y_{n-1}) = f(x_0, \ldots, y_n, \boldsymbol{P}) + \alpha \sum_{i=1}^{n} (l_i - l_0)^2 \tag{1}$$

where α is ...

$$f(x_0, .., y_n, P) = \frac{< P >}{2} \sum_{i=1}^{n} [\frac{l_i}{\boldsymbol{P}(x_{i-1}, y_{i-1}) + \epsilon} + \frac{l_i}{\boldsymbol{P}(x_i, y_i) + \epsilon}] \tag{2}$$

$$< P > = \frac{\boldsymbol{P}(x_0, y_0) + \boldsymbol{P}(x_t, y_t)}{2} \tag{3}$$

$$l_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{4}$$

$$l_0 = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2}/n \tag{5}$$

In order to minimize $L(\cdot)$, we use gradient descent method:

$$x_i^{t+1} = x_i^t - \eta \frac{\partial L(x_1^t, y_1^t, \ldots, x_{n-1}^t, y_{n-1}^t)}{\partial x_i} \tag{6}$$

$$y_i^{t+1} = y_i^t - \eta \frac{\partial L(x_1^t, y_1^t, \ldots, x_{n-1}^t, y_{n-1}^t)}{\partial y_i} \tag{7}$$

where η is the learning rate.

2 Calculate derivatives

Take derivative of $\partial L(\cdot)/\partial x_i$ as an example:

$$\frac{\partial L(\cdot)}{\partial x_i} = \frac{\partial f(x_0^t, \ldots, y_n^t, \boldsymbol{P})}{\partial x_i} + \frac{\alpha \partial \sum_{i=1}^{n} (l_i^t - l_0)^2}{\partial x_i} \tag{8}$$

For the first term (ignore constant $< P > /2$):

$$
\begin{aligned}
\frac{\partial f(x_0^t, \dots, y_n^t, \boldsymbol{P})}{\partial x_i} &= \frac{\partial \sum_{i=1}^{n} [\frac{l_i}{\boldsymbol{P}(x_{i-1}, y_{i-1})+\epsilon} + \frac{l_i}{\boldsymbol{P}(x_i, y_i)+\epsilon}]}{\partial x_i^t} \\
&= \frac{\partial \frac{l_i}{\boldsymbol{P}(x_{i-1}, y_{i-1})+\epsilon}}{\partial x_i} + \frac{\partial \frac{l_i}{\boldsymbol{P}(x_i, y_i)+\epsilon}}{\partial x_i} + \frac{\partial \frac{l_{i+1}}{\boldsymbol{P}(x_i, y_i)+\epsilon}}{\partial x_i} \\
&= \frac{1}{\boldsymbol{P}(x_{i-1}, y_{i-1})+\epsilon} \frac{\partial l_i}{\partial x_i} + \frac{1}{\boldsymbol{P}(x_i, y_i)+\epsilon} (\frac{\partial l_i}{\partial x_i} + \frac{\partial l_{i+1}}{\partial x_i}) - \frac{l_i + l_{i+1}}{(\boldsymbol{P}(x_i, y_i)+\epsilon)^2} \frac{\partial \boldsymbol{P}(x_i, y_i)}{\partial x_i}
\end{aligned}
$$

For the second term (ignore constant α):

$$\frac{\partial \sum_{i=1}^{n} (l_i^t - l_0)^2}{\partial x_i} = 2(l_i - l_0) \frac{\partial l_i}{\partial x_i} + 2(l_{i+1} - l_0) \frac{\partial l_{i+1}}{\partial x_i} \tag{9}$$

In total we have three derivatives to calculate:

$$\frac{\partial l_i}{\partial x_i} \qquad \frac{\partial l_{i+1}}{\partial x_i} \qquad \frac{\partial \boldsymbol{P}(x_i, y_i)}{\partial x_i}$$

The first two derivatives can be calculated by auto gradient in Pytorch, or using equations:

$$
\begin{aligned}
\frac{\partial l_i}{\partial x_i} &= [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2]^{-\frac{1}{2}} (x_i - x_{i-1}) \\
\frac{\partial l_{i+1}}{\partial x_i} &= [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2]^{-\frac{1}{2}} (x_i - x_{i+1})
\end{aligned}
$$

The third term can be calculated by Sobel kernel to go through the image $\boldsymbol{P}$:

$$\partial x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \partial y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{10}$$

my revision ends here.

For implementation, the terms can be represented as a vector:

$$\vec{l} = \begin{bmatrix} l_1 \\ l_2 \\ \dots \\ l_n \end{bmatrix} \qquad \vec{g} = \begin{bmatrix} g_1 \\ g_2 \\ \dots \\ g_n \end{bmatrix} \tag{11}$$

where:

$$g_i = \frac{1}{P(x_{i-1}, y_{i-1}) + \epsilon} + \frac{1}{P(x_i, y_i) + \epsilon} \tag{12}$$

Define the jacobian of $\vec{l}$ and $\vec{g}$:

$$J_x(\vec{l}) = \begin{bmatrix} \frac{\partial l_1}{\partial x_0} & \frac{\partial l_1}{\partial x_1} & \cdots & \frac{\partial l_1}{\partial x_n} \\ \frac{\partial l_2}{\partial x_0} & & & \\ \cdots & & & \\ \frac{\partial l_n}{\partial x_0} & & \cdots & \frac{\partial l_n}{\partial x_n} \end{bmatrix} \quad J_x(\vec{g}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_0} & \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_0} & & & \\ \cdots & & & \\ \frac{\partial g_n}{\partial x_0} & & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix} \tag{13}$$

Since g_i is only dependent on x_i and x_{i-1}, the jacobians for $\vec{l}$ and $\vec{g}$ will be:

$$J_x(\vec{\cdot}) = \begin{bmatrix} v1 & v2 & 0 & \cdots & \cdots & 0 \\ 0 & v1 & v2 & 0 & \cdots & 0 \\ \cdots & \cdots & & & & \cdots \\ 0 & & \cdots & & v1 & v2 \end{bmatrix} \tag{14}$$

Jacobian wrt. y is defined similarly.
As such, partial derivatives of the first term can be rewritten as:

$$\frac{\partial f(x_0^t, \ldots, y_n^t, \boldsymbol{P})}{\partial x_i} = J(\vec{g})^T \cdot \vec{l} + J(\vec{l})^T \cdot \vec{g} \tag{15}$$

Therefore, the gradient descent at iteration t:

$$\partial_i L(x_1^{(t)}, ..., y_{n-1}^{(t)}) = \partial_i f(x_0^{(t)}, .., y_n^{(t)}, P) + \alpha \partial_i [\sum_{i=1}^{n} (l_i - l_0)^2] \tag{16}$$

$$\frac{\partial v}{\partial t} = \boldsymbol{H}(x) \qquad \frac{\partial v}{\partial t} = \boldsymbol{NN}(x, v, \theta)$$