Image Processing for Bubble Detection in Microfluidics

Chen Fang Mechanical Engineering Department Stanford University

Introduction

Starting from recently years, microfluidics devices have been widely used to build the biomedical devices and micro fuel cells. In the research on microfludics, high speed camera is widely used to record the motion of liquid bubbles in the microchannel, based on which the dynamics of bubbles can be interpreted. However, the mechanical vibration of the system causes the channel to deviate from its original position during the image acquisition process. Figure 1(a) shows a typical image of a silicon microchannel with sidewall water injection, in which the injected water bubble and the skewness of the channel are evident. Hence, it is necessary to perform the skewness and translation correction on the raw images acquired by the high speed camera before the correct information of the bubble trajectory can be obtained from the images. Also, to extract the motion information from a huge number of images, we need to isolate the front bubble object from the background image, which will facilitate the automatic calculation of bubble velocity.

The spatial deviation of the image can be decomposed into angular and translational components. In the current project, we will first develop an algorithm to efficiently calculate the angular deviation of the image based on solving a minimization problem using Least Square SVM. Also, the translation correction of image can be accomplished in a straightforward way. After the realignment of the image is finished, the moving bubble can be separated from the background image by using Gaussian mixture model and EM algorithm to classify each pixel in the image as either the foreground or background. The test result shows that the present skewness correction algorithm works very well for the tilt angle less than 20 degree. Also, the background separation algorithm can successfully distinguish the moving object from the background image with still droplets in it for various illumination conditions. The major advantage of the current approach is that no iteration is required and computation is very cheap.

Skewness Correction

To correct for the angular deviation, we first convert the original gray-scale image (Fig.1(a)-4(a)) into binary image(Fig.1(b)-4(b)). Suppose the image tilt angle should be corrected by α , then the white pixel at (x, y) is moved

to (x', y'), where we have:

$$\begin{bmatrix} x'\\ y \end{bmatrix} = \begin{bmatrix} \theta'\\ \theta \end{bmatrix} \begin{bmatrix} x\\ y \end{bmatrix}, \quad \text{where} \quad \begin{bmatrix} \theta'\\ \theta \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha \end{bmatrix}$$
(1)

Since the biggest feature in the binary images is a pair of straight black lines representing the channel border, to implement the skewness correction, we only need to calculate the θ , such that the projection of the black pixels onto the y axis after the correction assumes the minimal value, i.e.:

$$\min \sum_{i=1}^{N} (y_i - \bar{y})^2 \quad \text{or} \quad \min \sum_{i=1}^{N} (\theta[x_i, y_i]^T - \overline{\theta[x, y]^T})^2$$
(2)

Here, we construct a training data set $\{X_i, Y_i\}$ corresponding to all the black pixels in the image, such that $X_i \in \mathbb{R}^2$

indicates the x, y coordinates of the pixel, and $Y_i \in R$ can be set as a constant. If we use $f(X) = \theta X + b, (\theta \neq 0)$ to

regress $\{X_i, Y_i\}$, then the regression error is :

$$\varepsilon = f(X) - Y = \theta X + b - Y \tag{3}$$

It is interesting to recognize that the skewness correction objective (2) can be expressed as:

$$\min \sum_{i=1}^{N} (\boldsymbol{\theta}[x_i, y_i]^T - \overline{\boldsymbol{\theta}[x, y]^T})^2$$

$$= \min \sum_{i=1}^{N} (\boldsymbol{\varepsilon}_i - \boldsymbol{b} - \boldsymbol{Y}_i - \overline{\boldsymbol{\varepsilon} - \boldsymbol{b} - \boldsymbol{Y}})^2$$

$$= \min \sum_{i=1}^{N} (\boldsymbol{\varepsilon}_i)^2$$
(4)

In deriving (4), we recognized that $Y_i \in R$ is set as a constant in our application.

By comparing (3) and (4), we find that to determine θ , it is equivalent to regress the training set $\{X_i, Y_i\}$ using the linear equation $f(X) = \theta X + b, (\theta \neq 0)$.

Here, we use the least square SVM to implement the regression. In particular, we consider the minimization problem:

$$\min \frac{1}{2} \theta \theta^{T} + \frac{1}{2} \zeta \sum_{i=1}^{n} e_{i}^{2}$$

$$s.t.Y_{i} - \theta X_{i} - b - e_{i} = 0$$
(5)

By constructing the Lagrangian function, taking derivative with respect to θ, b, e and Lagrangian multiplier a, we

have:

$$\begin{bmatrix} 0 & \mathbf{1}_{N}^{T} \\ \mathbf{1}_{N} & \boldsymbol{\Omega} - \boldsymbol{\gamma}^{-1} \boldsymbol{I}_{N} \end{bmatrix} \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{Y} \end{bmatrix}$$
(6)

with $Y = [y_1, \dots, y_N]^T$, $\mathbf{1}_N = [1, \dots, 1]^T$, $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N]^T$, I_N is an $N \times N$ identity matrix, and the kernel matrix

 $\Omega \in \mathbf{R}^{N \times N}$ with $\Omega_{ij} = x_i^T x_j$. Since the intercept b does not influence the $\boldsymbol{\theta}$, (6) can be simplified as:

$$\Omega a = \gamma^{-1} I a \tag{7}$$

Then non-zero vector *a* correspond to the first eigenvector of $\Omega \in \mathbf{R}^{N \times N}$.

Finally, the correction angular displacement $\theta = \sum_{i=1}^{N} a_i X_i^T$

After finishing the skewness correction, the next step is moving the image in X and Y direction to finally finish the image alignment. In the present case, such alignment can be achieved based on the position of the water injection point. We will not talk about this part in detail.

Moving Bubble Detection

To estimate the bubble trajectory in an automatic manner, we need moving bubble detection algorithm to separate the moving bubble from the channel and other still objects in the background for each frame in the video sequence. The easiest way to detect moving object in an image includes obtaining the background image first, and then subtract the background image from the original image, leaving the foreground moving objects. However, this technique is not applicable to the present study, since the background image capture process, making it very difficult to find a general background image applicable to all frames in the video. Here, we consider a single pixel and the distribution of its values over time. At a specific moment, a particular pixel may be either in the background state or in the foreground

state. Thus, the intensity value $i_{x,y}$ of a pixel (x, y) can be treated as the weighted sum of two Gaussian distributions:

distributions:

$$i_{x,y} = \phi_{background,x,y} g_{background,x,y} + (1 - \phi_{background,x,y}) g_{foreground,x,y}$$
(8)
where:

$$g_{background,x,y} \quad N(\mu_{background,x,y}, \sum_{background,x,y})$$

$$g_{foreground,x,y} \quad N(\mu_{foreground,x,y}, \sum_{foreground,x,y})$$

The model for pixel (x,y) is parameterized by the parameter

$$\boldsymbol{\theta}_{\boldsymbol{x},\boldsymbol{y}} = \{ \phi_{l,\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\mu}_{l,\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{l,\boldsymbol{x},\boldsymbol{y}} \mid l \in \{ foreground, background \} \}$$
(9)

Let i be the pixel intensity, L be a random variable indicating the label of the pixel in the image, our model defines the probability distribution:

$$P(L=l,I(x,y,t)=i \mid \theta) = \frac{\phi_{l,x,y}}{(2\pi)^{\frac{n}{2}} \mid \Sigma_{l,x,y} \mid^{\frac{1}{2}}} \exp(-\frac{1}{2}(i-\mu_{l,x,y})^{T} \Sigma_{l,x,y}^{-1}(i-\mu_{l,x,y}))$$
(10)

Knowing the parameter θ of the above probability distribution of each pixel, we can classify each pixel as either background or foreground based on the highest posteriori probability P(L = l | I(x, y, t)).

Here, our objective is to find parameters $\theta = \arg \max_{\theta} \prod_{t=1}^{T} P(L = l_t, I(x, y, t) | \theta)$, by taking derivative, it can be found

that:

$$\begin{split} \phi_{l,x,y} &= \frac{N_{l,x,y}}{T} \\ \mu_{l,x,y} &= \frac{M_{l,x,y}}{N_{l,x,y}} \\ \sum_{l,x,y} &= \frac{Z_{l,x,y}}{N_{l,x,y}} - \mu_{l,x,y}^T \mu_{l,x,y} \\ where \\ N_{l,x,y} &= \sum_{t=1}^T 1\{L_{x,y,t} = l\} \\ M_{l,x,y} &= \sum_{t=1}^T (1\{L_{x,y,t} = l\}I(x, y, t)) \\ Z_{l,x,y} &= \frac{\sum_{t=1}^T (1\{L_{x,y,t} = l\}I(x, y, t)I(x, y, t)^T)}{T} \end{split}$$

Since we do not have the labels $L_{x,y,t}$ for the training data, the above equations can not be calculated directly. Instead, we calculate a sequence of parameter settings, in which each setting is found by using the previous one to classify the data. In particular, suppose at the current step we have some distribution parameterized by $\theta_{l,x,y}$, we can use the expect

values of different labels according to $\theta_{l,x,y}$ as an estimate of their true value. Taking $N_{l,x,y}$ as an example, we have:

$$N_{l,x,y} = E[N_{l,x,y} | \theta_{l,x,y}] = \sum_{t=1}^{T} P(L_t = l | I(x, y, t), \theta_{l,x,y})$$
(11)

To classify each new frame, this approach requires calculating the summation over all the previous frames, which is quite expensive. As an alternative way, whenever we classify a new frame, we add its contribution to the current

statistics, which means that we are increasing our training set at each step, without reprocessing the previous frames in the training set. Hence, we have following algorithm:

Initialize $\theta_{l,x,y}$ for each pixel in the first image in the video.

For each new frame {

For each pixel in the new frame {

1. Update the parameter $\theta_{l,x,y}$ for mixture model:

$$N_{l,x,y} \coloneqq (1-\alpha)N_{l,x,y} + \alpha P(L_t = l \mid I(x, y, t), \theta_{l,x,y})$$

$$M_{l,x,y} \coloneqq (1-a)M_{l,x,y} + \alpha P(L_t = l \mid I(x, y, t), \theta_{l,x,y})I(x, y, t)$$

$$Z_{l,x,y} \coloneqq (1-a)Z_{l,x,y} + \alpha P(L_t = l \mid I(x, y, t), \theta_{l,x,y})I(x, y, t)I(x, y, t)^T$$

$$\theta_{l,x,y} = f(N_{l,x,y}, M_{l,x,y}, Z_{l,x,y})$$
(12)

2. Classify the pixel as background or foreground, based on the current mixture model

}

}

Since the illumination condition may change over the time, each frame's contribution to the background image need to be weighted according how far it is away from the current frame. Therefore, we introduce the relaxation factor α in (12). α lies between 0 and 1; The influence of the previous image on the current image decays exponentially with the distance. In the present study, we choose $\alpha = 0.5$.

Result

Figure 1 through Figure 4 shows the image processing result for four frames using the present algorithm for skew correction and moving bubble extraction. It is shown that the skewness correction method works very well for the images in various angles. In particular, the algorithm is not susceptible to the noise (scattered black pixels) in the binary image converted from the source image. Also, the moving bubble is successfully separated from the background channel. In particular, the algorithm correctly classifies the still bubble in the channel as background and excludes it from the resulted moving bubble image. By recognizing that such still object may exist at any place in the channel and may not be correctly removed by directly subtracting a predetermined background image from the source image, the advantage of our method is evident.

Summary

In the present study, we correct a sequence of skew images by solving a minimization problem based on LS SVM. The moving bubble object is extracted by classifying each pixel in the image using Gaussian mixture model. The result is quite satisfactory for the present application. The major advantage of the current methods is that no iteration is involved in both steps. In particular, the correction angle can be directly obtained by calculating the eigenvector of the resulted matrix, and the classification model for the pixels in the current frame can be updated by adding the contribution of the present image to the parameters of the previous image. Therefore, the computation is very cheap and an implementation in real time is possible.



Fig.1 image processing result for frame 1 ((a) original image, (b) binary image, (c) corrected image, (d) extracted moving bubble, note our algorithm can successfully remove the still bubble from the extracted image)



Fig.3 image processing result for frame 3