Laser Shadowgraph Image and Ultrasound Signal Feature Extraction Project

Gang Yao, Fangping Shi, Zheng Miao

College of Information Science and Electronic Engineering
Zhejiang University, China

Adviser: Prof. L. Jay Guo

Co-Adviser: Qiaochu Li, Ph.D

Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor

September 30th, 2018

Abstract

Ultrasound Bubble Extraction

In photo-acoustic study, focused ultrasound waves in water could contain energy per cubic large enough to tore the water molecules apart and generate vacuum bubbles in the water. This phenomenon has always been mysterious. It is expected to be widely studies for its potential of application in non-invasive surgery. Laser shadowgraphy imaging is used to capture the those bubbles. To better understand the mechanism in the formation of these bubbles, we need to count the number and area of the bubbles in a set period of time. However, this counting task could be far too demanding for human inspectors. To resolve this dilemma, this project implement an adapted K-NN(K nearest neighbor) background segmentation algorithm for bubble detection using Python and OpenCV. This widely used machine learning approach yielded satisfying result for our specific scenario, with the small bubble detection accuracy over 90% and big bubble detection accuracy over 85%. Also, we brought forward a new algorithm tackling the problem of counting overlapping bubbles. Traditional circle detection algorithm is typically based on Hough circle transformation and is not suitable for noisy or overlapping circles. Our solution, on the other hand, solves the problem with a fast and robust algorithm that can detect and count the numbers and positions of overlapping circles with high accuracy using multiple techniques. The application of this algorithm is not confined in bubble counting in supersonic shadowgraph only. It can also be applied to many other fields such as automatic cell counting in medical field.

Ultrasound Signal Recovery

In monitor the cavitation behavior with fiber hydrophone ultrasound detector, the signal energy level is almost identical to that of the white noise. This makes it hard to locate and see the actual position and shape of the signal. We implement the wavelet transform to investigate the frequency and spatial domain position of the signal. Then, with multiple level of noise filtering, the useful signal can be extracted and showed in real-time. The tested lag is estimated to be less than one second.

Contents

List of Figures

1 Background

Photo-acoustic effect is the formation of acoustic waves upon the absorption of modulated or pulsed light mainly through thermoplastic expansion. It has been widely applied in medical imaging, spectroscopy, photo-acoustic spectrum analysis, defect detection and therapeutic treatment. By implementing a highly-efficient photo-acoustic generation layer made from candle soot/PDMS composite and deposit onto a concave BK7 glass lens, we have developed a self-focusing PA lens that can focus ultrasound waves in a ellipsoid of 90m(minor axis) by 200m(major axis). With an input laser fluence of 7.5mJ/cm2, the focused region can reach a negative pressure of 40MPa, which exceeds the cavitation thresholds in water in free space. Therefore, we have demonstrated controlled cavitation behavior in a submillimeter region, and it has been applied in soft-tissue ablation, nozzle-free jetting and needle-free injection. The cavitation bubbles are monitored simultaneously by a shadowgraph imaging system excited by laser. In order to study the cavitation probability under different pressure, we propose to apply a adapted KNN background segmentation method for bubble counting in the shadowgraph images, the method is based on transient and dark feature of the bubbles.

Instead of imaging guidance by laser shadowgraph, we also monitor the cavitation behavior with fiber hydrophone ultrasound detector. The advantage of FP is the high damage threshold so that it can measure signals up to hundreds of MPa, which will be good enough for measure any intense focused pressure fields or shock waves. However, the sensitivity of the system is much limited and the SNR is low without averaging, therefore, limiting the application for bubble signal detection. We propose here a wavelet transformation method to locate the time and spatial location of the interested signal.

This study will not only help better understanding the cavitation threshold at different pressure, but also help us to compare the cavitation probability in different condition more accurately. At the same time, with the noise elimination algorithm, we are able to increase the SNR of the FP detector in 3 times,

2 Object Specification

2.1 Laser shadowgraphy - Detect & Count Bubbles

Laser shadowgraphy images are recorded as pseudocolor RGB frames. In order to count the total number of bubbles generated, the first step is to extract and segment the regions of interest (ROI). Any ellipse detection algorithm won't work in our case due to the limited resolution of the camera and small size of the bubble. Therefore, we decided to take advantage of two key features of the bubble: color and transiency.

In lab practice, shadowgraph videos are commonly recorded at 12 frames per second (about 80 ms period). There is fundamental difference in small bubbles with occasional occurrence and large bubbles that endures for long time.

For the small bubbles, their life cycle is at a comparable level to the frame rate. Hence, bubbles appear and disappear quickly in the video. Consider this "transient" nature of bubbles, we implemented a background segmentation algorithm as the first step of feature extraction. Next, to separate featured bubbles in the foreground, we applied morphological operation and black color segmentation multiple times to extract them.

However, for the large bubbles, their life cycle is longer and there is the problem as overlapping may occur. So it would require an additional step to count exactly how many bubbles are actually overlapping.

Chapter 3 discusses how we extract the desired feature - bubbles from the recorded laser shadowgraphy image. Small bubbles are directed counted and big or overlap bubbles are sent to the overlapping bubble counting algorithm discussed in Chapter 4 to decide exactly how many bubbles are overlapped. There results are combined to yield a resulting diagram showing bubble number and size as illustrated in Figure 13 and Figure 12.

2.2 Ultrasound Waves - Denoise & Recover Signal

Because the invisibility of ultrasonic wave, there is no direct method to see ultrasonic wave. In our experiment, we placed a laser generator in the one side of the water tank and we placed a light sensor in another side of the water tank. The laser generator would emit laser toward the water and the light sensor would generate electric signal whose frequency and amplitude are relative to light property. When there occurs waves in the water, the water refractive index will change which cause light intensity changed. So The light intensity signal measured by light sensor can reflect some characters of ultrasonic wave.

Previous methods involve taking average of large number of signal samples. However, this approach is time-consuming, painstaking and hard to adjust. This obstacle inspires us to bring about a new method to denoise the signal within several samples using the knowledge from signal processing and wavelet transform. We also aim to embed our matlab code into the original experiment terminal so that the inspector can see the recovered clear signal in real time.

3 Bubble Detection and Segmentation

The most distinctive feature that we utilize for bubble detection is their transient existence. By modeling the relatively stable background of a video stream, we obtain the foreground mask. The mask consists of three types of pixels: background, foreground and shadow. When applied on the original image, pixels whose position on the mask is background are set to pure black while those whose position on the mask is foreground remain unchanged - shadows are sorted for different conditions. It works as though we used a "mask" to segment the regions of interest.

General Structure

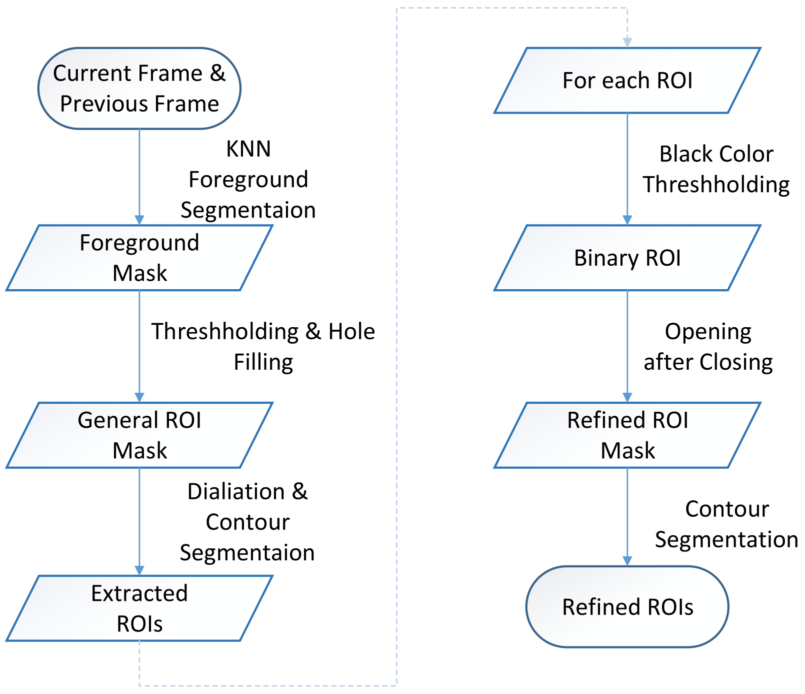
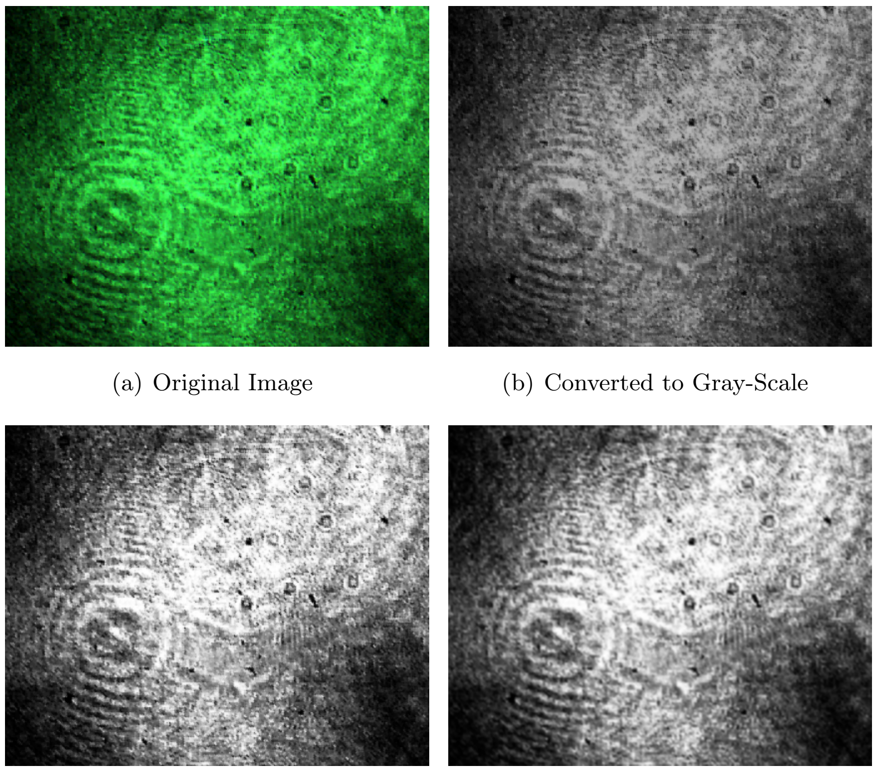

Current Frame & Previous Frame
KNN Foreground Segmentaion
Foreground Mask
Threshholding & Hole Filling
General ROI Mask
Dialiation & Contour Segmentaion
Extracted ROIs
For each ROI
Black Color Threshholding
Binary ROI
Opening after Closing
Refined ROI Mask
Contour Segmentation
Refined ROIs

Figure 1: Foreground Detection General Structure

Before segmenting the bubble, we first took several per-processing techniques:

- Convert to Gray Scale The original colored shadowgraph is captured with green laser reflected and refracted in the water box, and thus only have the color green. We convert it to 8-bit gray-scale so that we only need to deal with one dimension of data.
- Histogram Equalization To minimize the influence of light intensity change, we apply histogram equalization. An ideal equalization would result in the even distribution of number of pixels on all the gray-scale levels, increase contrast and above all sharpness.
- Gaussian Blur Background segmentation is sensitive to small noise. So we hope to best reduce some random noise in the pre-processing phase. There could be the trade off as to blur the image, but as long as this is contained in acceptable level, blurriness won't effect detection.


(a) Original Image (b) Converted to Gray-Scale

(c) After Histogram Equalization (d) After Gaussian Blur

Figure 2: Result of Preprocessing

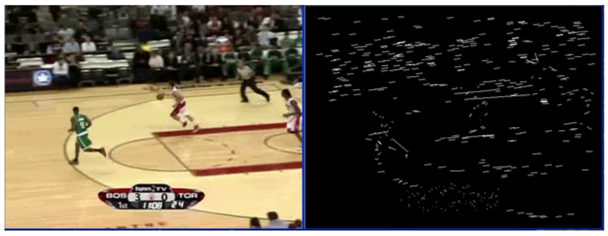
3.1 Foreground Segmentation Overview

Foreground detection is among the most widely discussed issues in image processing and computer vision, often serves as the preprocessing step for higher leveled tasks. [1]Many of the practical applications do not take interest in the entire image, but rather focus on certain objects in the image. Foreground detection aims to extract the information of change by segmenting the region of interest (ROI) from the original image. Usually, a sequence of video frames is needed as the input, and the segmented ROI as the output.

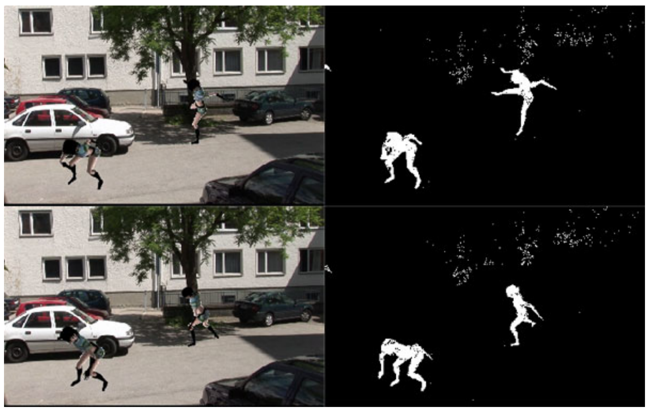
The major challenge of foreground detection system include:

- vibration or small deviance introduced by the camera
- lighting intensity change, random noise and periodic movements
  e.g. waves and shadows and long term slow changes.
- ghosting effect and halos

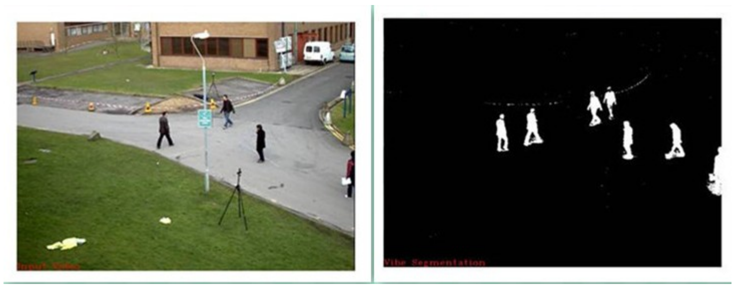
Any foreground detection techniques involve the modeling of background. By evaluating the difference of the input image to the background, the foreground information is extracted. However, to rigorously define and establish the background image can be challenging, especially when there is strong variation in the color, intensity, shadows, quaility, nosie, interior and exterior etc. in the image stream. Scenarios where these techniques apply tend to be diverse. Systems need to be able to adapt to these changes.



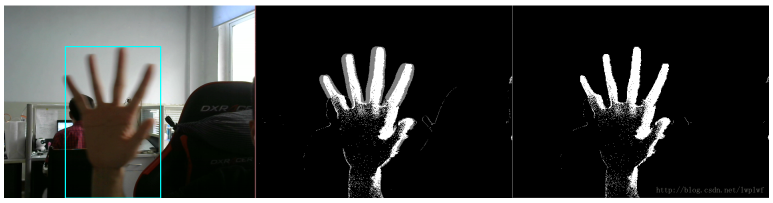
(a) Scattered Detection caused by drastic camera movement



(b) False detection of trees on the background



(c) Ghosting effect behind the human figure



(d) Halos in the center of an moving hand

Figure 3: Typical Failures of Foreground Detection[2][3]

3.2 Previous Works and Ideas

The background can be featured as all the pixels of the objects in the scene that is relatively still compared to the foreground pixels, which are relatively versatile. The key to foreground detection is to measure and distinguish between the extent of the variation on different positions. Here are the few fundamental ideas that lays the corner stone for any advanced technique.

3.2.1 Background Segmentation

The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model".

$$F(t) = I(t) - B \tag{1}$$

whereas the $F(t)$ denotes the foreground of the current frame, $I(t)$ denotes the original image and B denotes the set background model.

The benefit of background subtracting is that it will never cause ghosting and also effectively extract any change in the scene. But this approach is only effective when we have already known the background, the camera is still and the scene is not prone to any change within a set of time.

3.2.2 Inter Frame Subtraction

This method subtract from the current frame with previous frame and then apply binary threshold segmentation to extract the difference.

$$difference(t) = \begin{cases} 1 & f(t) - f(t - t_0) \geq thresh \\ 0 & f(t) - f(t - t_0) < thresh \end{cases} \tag{2}$$

Usually, the gap t_0 is set to 1. However, subtracting with neighboring frame increase the likelihood of introducing ghosting effect. Although this can be easily eliminated by subtracting with more previous frames, to define the exact gap between those two can be demanding, since there is the trade off that subtracting distant frames (increasing t_0) reduce ghosting effect but is also likely to miss out real foreground.

The advantage of inter frame subtraction is that the idea is straight forward and it is not prone to the variation in lighting conditions. However, the more conspicuous downsides including sensitivity to camera motion, failure in detecting still or slow object and the difficulty in choosing threshold value all make it not applicable in actual situations.

3.2.3 Integrate the Ideas - Adaptive Method

The problem with naive background subtracting is that with a stale background model, should any change take place in the current frame, the difference will be wrongly detected as foreground. Hence, a method dynamically updating the background model must be introduced to adapt to environmental change. Noticing that for inter frame subtracting, there is very good performance for adaptive change, we utilize the idea to update the background model in a given amount of time.

To know when and where, also how long the period is to calculate and update the new background, many methods introduce probabilistic models for evaluating the density function of one particular pixel. Some of the recent efforts A pixel in the current frame is regarded as a background pixel if its new value fits into its density function. The next step would be to estimate appropriate values for the variances of the pixel intensity levels from the image. Some of the recent efforts furthered the idea, such as belief propagation algorithm[4] or Histogram difference modeling[5].

The most widely adopted one is the Gaussian mixture model (GMM) proposed for background subtracting[6] and efficient update equations were given[7]. GMM has been improved by many previous works, with added hysteresis threshold[8] and tilt calibration[9], however, after implementing the so far the most comprehensive GMM approach, we found that it performed poorly. The reason could include the strong background varying noise which make the hypothesized Gaussian threshold not so accurate. And thus we turned to another rather innovative idea utilizing the K-Nearest-Neighbor(KNN) unsupervised learning algorithm structure.[10]

3.3 KNN Foreground Segmentation Algorithm

The value of a pixel at time t in gray-scale is denoted by $x^(t)$. The pixel based background subtraction involves decision if the pixel belongs to the background (BG) or foreground object (FG). The pixel is more likely to belong to the background if

$$\frac{p(\mathrm{B}G|x^{(t)})}{p(\mathrm{F}G|x^{(t)})} = \frac{p(x^{(t)}|\mathrm{B}G)p(\mathrm{B}G)}{p(x^{(t)}|\mathrm{B}G)p(\mathrm{F}G)} \tag{3}$$

is larger than 1 and vice versa. Most modern day techniques assume uniform distribution for the appearance of the foreground object $p(x^{(t)}|\mathrm{FG})$, because we can never know when and how long will the foreground object persist. KNN segmentation also had this supposition.

But Unlike the GMM approach which presuppose the background pixel density function into Gaussian distribution, KNN segmentation start by using a uniform kernel to estimate the density function and dynamically update it.

3.3.1 Introducing KNN

KNN is one of the most basic unsupervised learning algorithms for classification. It is commonly used for its easy interpretation and low calculation time. The objective of any classification model is to sort one target into its corresponding class. Figure 4 help illustrate how KNN works to fulfil its sorting task.

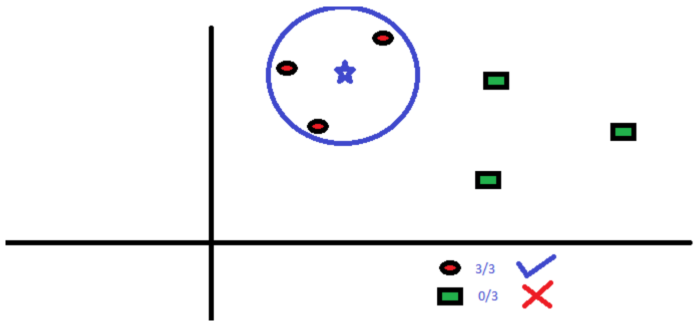


Figure 4: How KNN Works[11]

The shapes are sorted into two classes based on two attributes: x and y coordinate, namely circles and squares. Now a shape unknown of its class awaits to be classified as either circle or square. KNN takes the so called “voting” mechanism. The nearest K samples are eligible to vote, and hence the name KNN. Circles would try to vote the star to be a circle, while squares vote it to be a square. Sometimes every vote is timed by a factor, the so called "significance" of the vote. And by summing up all the votes, we reach a conclusion. In this case, 3 nearest neighbors voted circle, therefore the star is sorted into the class circle. The choice of the parameter K is vital to the correct classification.

3.3.2 Time Domain KNN Classification

To sort all the pixels $x^{(t)}$ of the current frame $f(\mathbf{x}_p, t)$ - $\mathbf{x}_p$ denotes the position of the pixel - into either class FG or BG. The first step is to compare the intensity level $x^{(t)}$ with k_t other pixels on the same position of the previous frames.

Estimating Background Probabilistic Density Function

Each of the history pixel $x^{(t-t_k)}$ here has an distinctive probabilistic density function for the potential background. Density estimation using a uniform kernel start by counting the number of sample k_t from the set ζ: $\{x^{(t-t_k)} \mid\mid t_k \in [1 : k_t]\}$ that lies within the threshold distance D on intensity level (gray-scale 0 255). The estimate is given by

$$\hat{p}(\mathbf{x}|\zeta, BG) = \frac{1}{k_t} \sum_{t-k_t}^{t} \xi \left(\frac{|x^{(t-t_k)} - x^{(t)}|}{D} \right), \ where \quad t_k \in [1 : k_t] \tag{4}$$

the kernal function here is:

$$\xi(u) = \begin{cases} 1 & u < thresh \\ 0 & u \geq thresh \end{cases} \tag{5}$$

Typically, the thresh here is set to 0.5. In practice, the purpose of the kernel function is only to smooth the estimate, but the gray-scale threshold distance D is critical. Elgammal[12] proposed using $D = med/(0.68\sqrt{2})$, where the med here denotes the median value of $|x^{(t-t_k)} - x^{(t)}|$ from ζ.

Fixed kernel size D for the whole density function might not be the best choice. The so called "balloon estimator" adapts the kernel size at each estimation point $\mathbf{x}$. Instead of trying to find the globally optimal D, we could increase the width D of the kernel for each new point $\mathbf{x}$ until a fixed amount of data k is covered. In this way we get large kernels in the areas with a small number of samples and smaller kernels in the densely populated areas. The balloon estimate is often used for classification problems since it is related to the kNN classification. One nearest neighbor is common but to be more robust to outliers we use $k = [0.1 * k_t]$.

Voting On the Current pixel

After obtaining the density function for background, we are now able to compare the posibility between background and foreground, since the distribution of foreground is uniform. A pixel is sorted into the background if

$$\begin{aligned} &\hat{p}(\mathbf{x}|BG) > P_{thr} \\ &where \quad P_{thr} = \frac{p(\mathbf{x}|FG)p(FG)}{p(BG)} \end{aligned} \tag{6}$$

$p(FG)$ and $p(BG)$ are calculated via the average result of the previous k_t samples.

$$p(BG/FG) = \frac{1}{k_t} \sum_{t-k_t}^{t} \hat{p}(\mathbf{x}^{(\mathbf{t}-\mathbf{t_k})}|BG/FG) \tag{7}$$

Figure 5 shows an intuitive look of how the current pixel is to be judged by previous votes, although we have added much probabilistic model to achieve more adaptive performance, the basic idea coinside with naive k-NN.

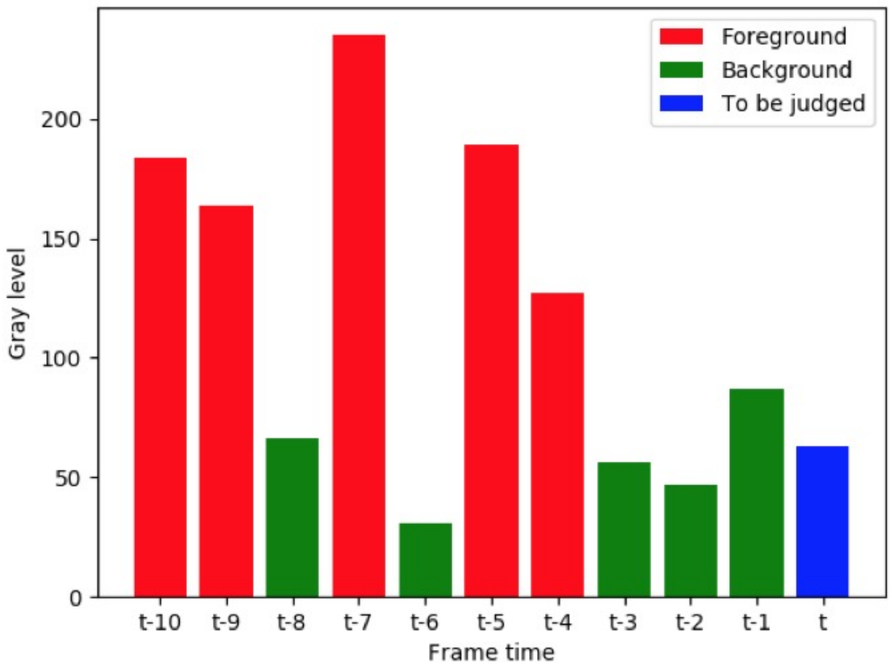


Figure 5: Time Domain Classification

$$f(\mathbf{x}_p, t)|x_{p0}: \quad \text{set } \zeta: \ \{x^{(t-t_k)} \ || \ t_k \in [1:k_t]\} \ \sim \ x^{(t)} \tag{8}$$

The x-axis shows each individual pixel on the same position at different time. The y-axis shows the intensity level of that particular pixel on gray-scale. The red rectangles in Figure 5 shows that the pixel was previous sorted into foreground and green rectangles sorted into background. After applying the mentioned math, the blue pixel will be classified as background, also matches the intuitive feeling that it is "most similar" to the previous background pixels.[8 pt] This step perfectly eliminates "ghosting effect" while stay robust in detecting any movement thanks to the probabilistic approach and dynamic update strategy.

3.3.3 Spatial Domain KNN Classification

This step serves to check the result of time domain classification. It takes into account K pixels that is no farther than a threshold distance to the current pixel, and allow them to vote, whose voting weight correspond to their distance to the pixel.

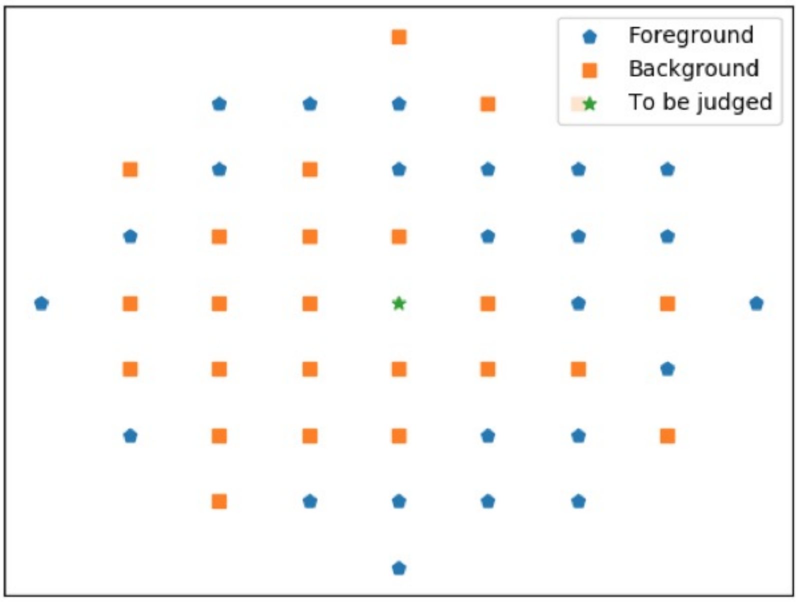


Figure 6: Spatial Domain Classification

Suppose the center pixel **c** has k neighboring pixels n_i whose Euclidean distance to the center is d_i. A fixed distance thresh D_{th} is set to enclose k samples to vote. The result is summed up as

$$C_{check} = \frac{1}{k}\sum_{i=0}^{k} tag_k \cdot N(\mu, \sigma^2) \tag{9}$$

where *tag* is the classification result of the previous step. 1 if foreground and -1 if background. The *new tag* is set to 1(foreground) if $C_{check} > 0$ and 0(background) if $C_{check} \leq 0$.

If *new tag* result coincide with its previous *tag*, then the class is accepted and background model updated. If it doesn't, the pixel is then sent into shadow detection function, which decides whether it belongs to the class shadow according to the extent of variance from its precedent pixel.

The spatial classification step help in increasing the robustness against slow moving objects and periodic movement. By increasing the threshold distance and including more spatial samples, the model would be more stable not to segment out less drastic changes but is also likely to miss out real bubble. In practice, this value ought to adjusted to different applications.

3.3.4 Obtaining Foreground Mask

After all these two steps, we obtain the original version of the foreground mask. The dark pixels(0) denotes background, gray pixels(127) denotes shadow and white pixels(255) denotes foreground.

Figure 7 shows the typical result of detection with small bubbles. Small bubbles survives for less than the period of frame rate. Therefore, it is unlikely that there will be any individual bubbles appearing in two consecutive frames.

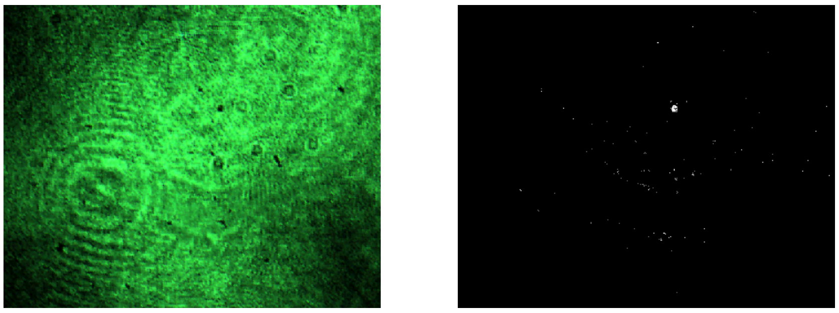


Figure 7: Raw Foreground Mask with Small Bubbles

However, for large bubbles, they survive much longer and the possibility of overlapping increases, especially on the focal point. For this reason, the area inside the overlapping large bubbles stays dark for a long time while the outer areas experiencing morphological change, leading the model to falsely predict these pixels into the background.

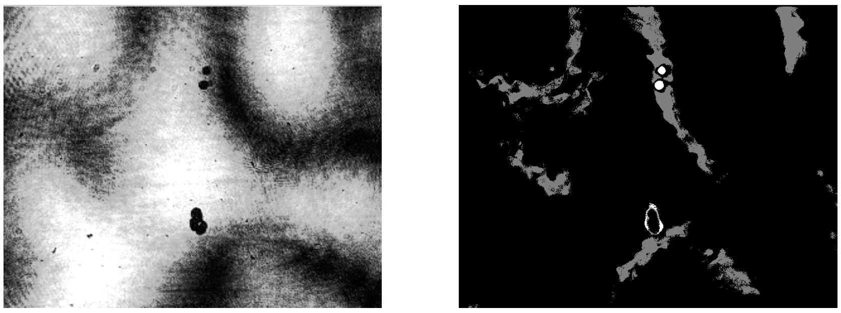


Figure 8: Original Image in Gray-scale and Raw Foreground Mask

Post-processing is required to

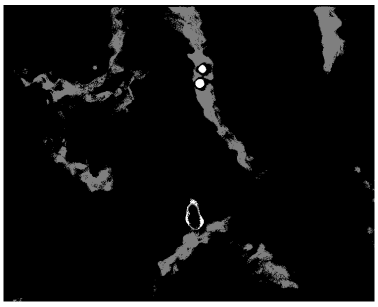
- eliminate grained noise and decide on the shadow pixels
- fill up the holes inside bubble contour area

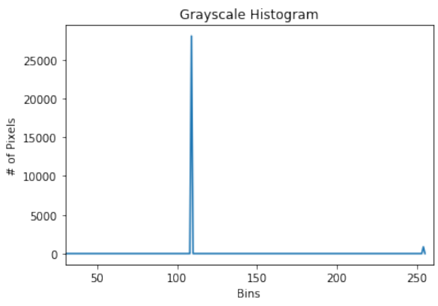
3.4 Post-processing

3.4.1 Shadow Threshold

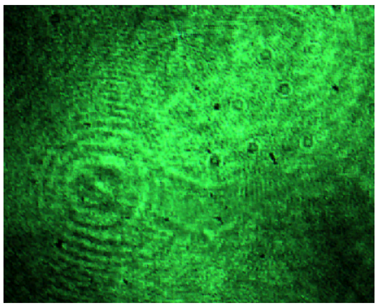
The reason why pixels are sorted into shadow is that when strong deviations occur regionally, these regioned will be classified as foreground in time domain k-NN classification. However, in spatial domain k-NN classification, the model found those pixels to be more related to their background neighbors. When time and spatial domain results do not concur, the dispute is logged by classify these pixels as shadows.

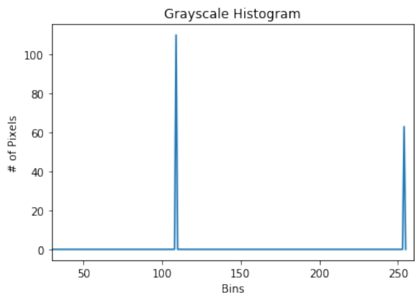
To decide on the fate of these shadow pixels, we calculate the number of each type of pixels, if the number of shadow is relatively small compared with foreground pixels, then they are merged into the foreground (as in Figure 9-ab). Else if their number far exceeds that of the foreground, they are then merged into the background (as in Figure 9-cd).


(a) Strong Shadow



(b) Strong Shadow Histogram


(c) Light Shadow Histogram



(d) Light Shadow Histogram

Figure 9: Shadow-Foreground Histogram

3.4.2 Morphological Operations

Erosion and Dilation

Erosion operation erode away the boundaries of foreground object(white). The kernel (an odd number n*m matrix of shape cross, rectangle or ellipse) slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is made to zero.

Dilation is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object. [13]

Closing and Opening

Opening is just another name of erosion followed by dilation. It is useful in removing noise.

Closing is for Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.[8 pt] We perform two iterations of Opening operation by a (5x5) elliptical kernel and then two iterations of Closing Operation by a (9*9) elliptical kernel on the raw mask, whose result yields Figure 10 with all the small holes filled and exterior noise filtered.

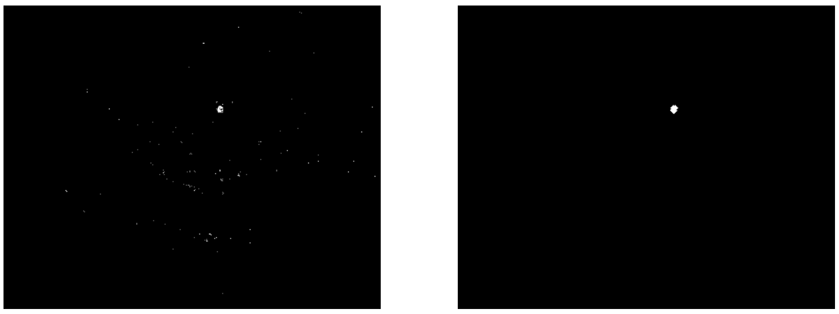


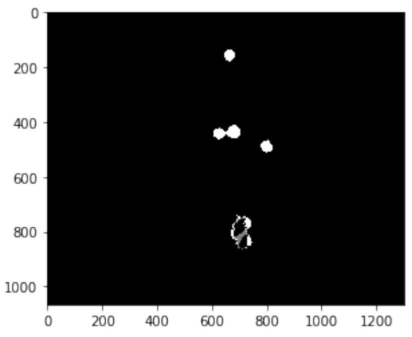
Figure 10: Before and after Morphological Operation

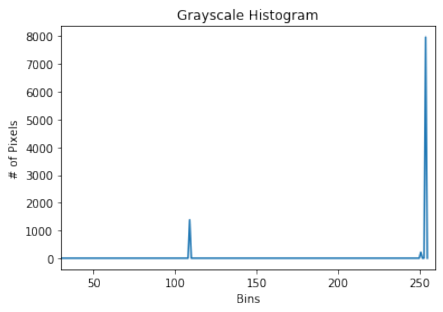
3.4.3 Flood-fill

This step is only for images containing big overlapping bubbles.

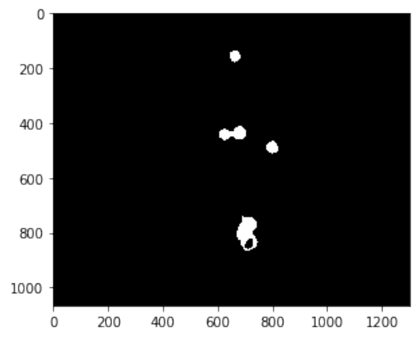
To fill up the big holes, we apply the "floodfill" method. The word vividly describes how this works: First, starting from the right up corner we start to "flood" the whole image - the initial point can be seen as the root, from the root, a search is started in three directions, namely right, down and right/down. Using depth-first-search(DFS) we go through all the points on the image plane, setting the locations where the the original mask image is '0' to '1'(or 255). The DFS algorithm will not go into any node that is featured '1'(or 255) on the original mask. - We can view the white areas on the original mask as "hign land" and with a flood coming through, all the high land and protected basin will not be affected. Then when we bit wise inverse the flooded region, high land and basins are extracted. Adding back this extracted region to the original mask, any holes (or basins) will be filled up.

Noticing that the contours of the overlapping bubbles may not be connected. We first dilate the raw mask by 3*3 square then perform floodfill, and repeat "dilate then floodfill" until there is no more holes to fill. Of course, dilation distort the original image. So after floodfill, we also reconsider the segmented contour region on the original gray-scale image will binary threshold. This way, the original shape of the bubble can be reserved.


(a) Raw Mask



(b) Shadow-Foreground Histogram


(c) Floodfill 1st Iteration

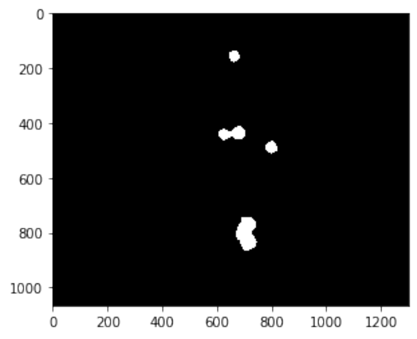

(d) Floodfill 2nt Iteration

Figure 11: Flood-fill Demonstration

3.5 Performance Analysis

- **Small bubbles with occasional occurrence**

  Test subject is 276 frames with only 8 frames (human count) containing one bubble. There is strong intensity variation influences. The test result fits into the human count result as is shown in Figure 12. x-axis numbers the bubble, y-axis is the size of the bubble. However, possible limitations include:

  First, strong intensity fluctuation could result in scattered foreground mask. These frames are typically discarded, but if bubble appear in these frames, it will be missed.

Table 1: Small Bubble Count Performance

Human Count	Computer Count	Percent Match
77	89	84.42%
32	33	97.87%
17	19	88.23%
124	126	98.39%

Second, if the bubble size is similar to that of a dark noise, it will also not be detected. However, human inspection is also expected to fail should this occur.

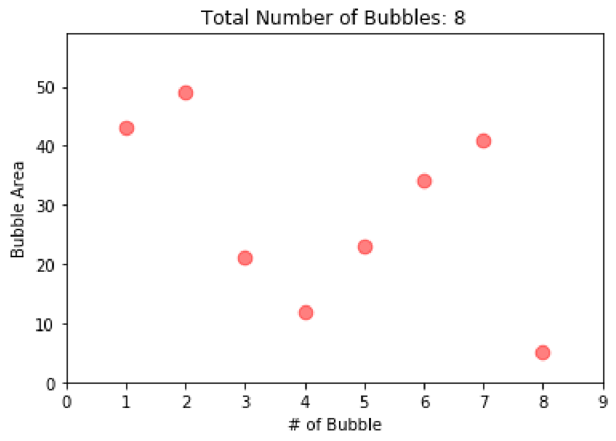


Figure 12: Result for Small bubbles with occasional occurrence

Other test result is listed in Table 1. Human count could also be reliable, but for now we assume that human count result is the "ground truth". In this way, the average percent match is 92.23%.

- **Big bubbles with rapid occurrence - Overlapping**

  Test subject is 335 frames with 1-7 bubbles appearing in every frame. Human eyes will not be able to count all of them. Figure **??** demonstrates the counting result with no overlapping check. Any contour area segmented are treated as one individual bubble.

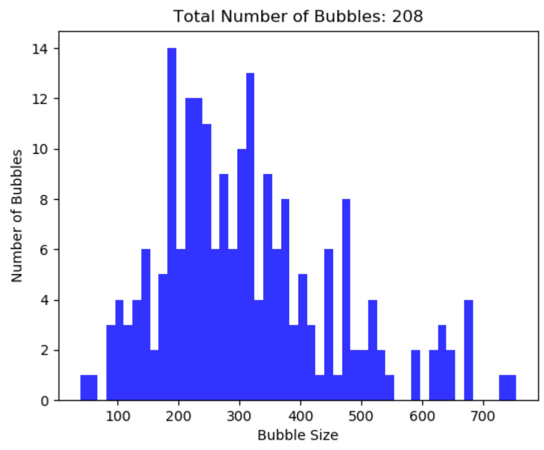


Figure 13: Result for Big bubbles - No Overlapping Check

With overlapping check, segmented contours are sent to Chapter 4 for fitting. The final counting result for the subject video is shown in Figure 14.

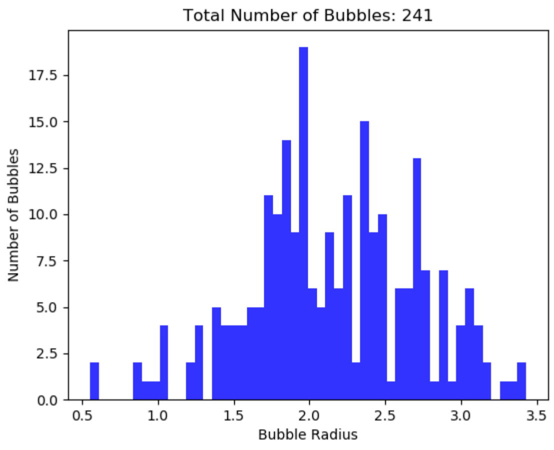


Figure 14: Result for Big bubbles - Overlapping Check

Precision with counting rapidly appearing big bubbles cannot be fully evaluated since it is simply a far too demanding task for human eyes. But by looking into the debug detection images, almost all the bubbles are successfully detected and segmented. We conservatively estimate the counting precision to be over 80

4 Overlapping Bubble Counting Algorithm

After bubble foreground is extracted from original images, we are able to count the total numbers, positions and areas of bubble for the subsequent analysis.

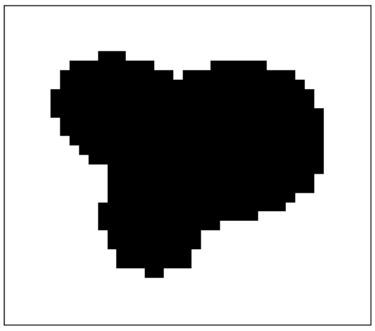


Figure 15: Detected Bubble Example

4.1 Previous Work

Traditional circle detection involves circle Hough Transform (CHT). The circle Hough Transform is a feature extraction technique for detecting circles.[14] It is a specialization of Hough Transform. The circle candidates are produced by "voting" in the Hough parameter space and then select the local maxima in a so-called accumulator matrix.[15]

In a two-dimensional space, a circle can be described by

$$(x - a)^2 + (y - b)^2 = r^2 \tag{10}$$

where (a, b) is the center of the circle, and r is the radius. If a 2D point (a, b) is fixed, then the parameters can be found according to (10). The parameter space would be three dimensional, (a, b, r). And all the parameters that satisfy (x, y) would lie on the surface of an inverted right-angled cone whose apex is at $(x, y, 0)$. In the 3D space, the circle parameters can be identified by the intersection of many conic surfaces that are defined by points on the 2D circle. This process can be divided into two stages. The first stage is fixing radius then find the optimal center of circles in a 2D parameter space. The second stage is to find the optimal radius in a one-dimensional parameter space.

Hough Transform is widely used in circle detection. However, it only works well on nearly-perfect circles without too much noises, which is not the case in this project.

4.2 Major Problems

The major challenges we encounter in bubble counting are

- overlapping phenomenon
- non-circular shape
- boundary noise and zigzags

Overlapping is very common between bubbles. We have to infer the centers and radii from arcs of the circles that overlap with each other.

Another important problem that undermines the effects of circle detection is that the bubbles are hardly circular. Both small-scale noises such as zigzags on the boundary and large scales of noises that contort the circular shape of the bubble are everywhere, which makes it extremely hard to perfectly fit a bubble with circle.

4.3 Algorithm Overview

We come up with a new algorithm that tackles the problem. It uses part of the circle, namely arc, to infer the position of center and length of radius. To make things clear, we will use program generated overlapping circle image to explain the algorithm.

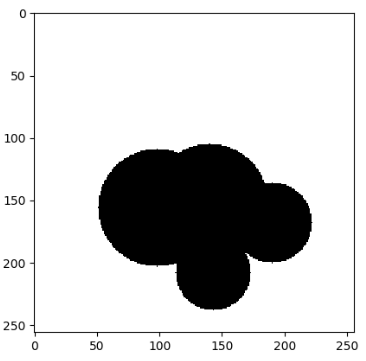


Figure 16: Generated Overlapping Circle Image

4.3.1 Boundary Extraction

The first step of the algorithm will be extracting the boundary of foreground. Many useful boundary extracting algorithm can be applied here, including erosion or more complicated Canny edge detection. Considering the features of the bubbles, which are filled black circles with white background, there is no disturbing gray level information that is common in general pictures. As a result, simple erosion can perfectly extract the boundary and there is no need to suffer the time-consuming Canny detector.

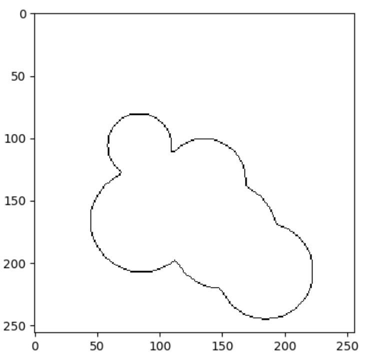


Figure 17: Extracted Boundary

4.3.2 Feature Points Extraction

After we have the boundary of the bubbles, we are interested in the feature points on the boundary. Feature points are those points that have strong gradient variation and are likely to be the intersection points of different circles.

Boundary Points Sorting

For gradient calculation, we first need a way to sort the boundary points in clockwise or anti-clockwise order.

Consider the 8-neighborhood of a certain pixel, we want to find the next pixel in the points sequence based on the current and previous pixel, which we have knowledge about. We define two directions. The direction that the boundary "comes from", denoted by $D1$, and the direction that the boundary "goes to", denoted by $D2$.

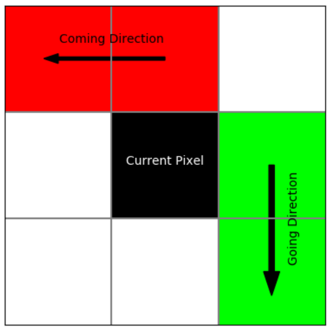


Figure 18: Direction of Previous and Next Pixel

In both $D1$ and $D2$, there are at most 2 points in the direction, which are the one on the 4-neighborhood (edge points) and the one on the 8-neighborhood (corner points). It is important to notice that if there are 2 points in $D1$ or $D2$ direction, then the next or previous point must lie in the 4-neighborhood and we should not take the point in the 8-neighborhood into account. If there is only one point in $D1$ or $D2$ direction, we will just take that very point as previous or next point.

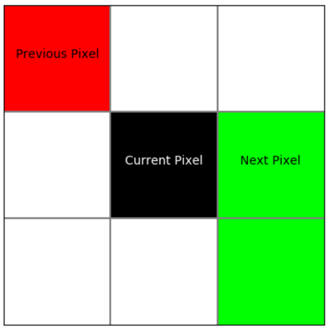


Figure 19: Positions of Previous and Next Pixel

After we have this analysis, we can do the points sequence sorting. We first find the leftmost point in the boundary (take the lowest point if there are multiple leftmost points) as the initial point. At each time, we check the 8-neighborhood of the current point. Since we know the previous point we find, we have the knowledge of the $D1$ direction. We remove all the points (1 or 2) in $D1$ direction and the points left are on the $D2$ direction. We then check the number of the points on the $D2$ points. If there are 2 points on the direction, we take the one on the 4-neighborhood as next point and if there is only 1 point, we directly take it as the next point. We iterate this algorithm the initial point is encountered again. When the algorithm stops, we will have a sorted boundary sequence, which will help us do the gradient calculation.

Gradient calculation

When we have the sorted boundary sequence, we should calculate the gradient. Since in real laser shadowgraph, the boundary contains strong noise like zigzag, it is unwise to directly calculate the gradient based on points position.

We come up with 2 ways to calculate the gradient. The first way is to take average gradient among K nearest points. This helps reduce the noise, but it requires much computation. The second way is to sample the boundary points such that only 1 point in every N continuous points in the sequence is taken into account. This reduces both the noise and the computational complexity, but it may lead to some information loss. In practice, we use the latter one. Also, in practice, to avoid the information loss, we can change the sampling step size and do multiple sampling, then we can take the best result.

After we have the gradient, we have to set a criterion to decide whether some points are feature points and others are not. This can be done by find the left gradient and right gradient of a certain point and take difference. In ideal mathematical analysis, if a curve is smooth, say, it is first-order differentiable, then the left derivative should be equal to right derivative. But in this case, since we only make approximations to the direction of both sides, the differencing method works. The steps are simple. We just calculate the gradient using the methods mentioned in previous part and we have the gradient in one direction, say, clockwise. Then we reverse the whole point sequence, and calculate the gradient again. This time we have the anti-clockwise gradient. After we have the gradient of both directions, we can make difference, take absolute value and calculate the corresponding angle on this point. If the angle is near 0 or near π, then we consider this point "smooth", which means this point is not likely to be a feature point. And if the angle is near $\pi/2$, then is point is likely to be a feature point that lies on the intersection of 2 circles.

4.3.3 Arc Extraction

After we have the feature points, we can extract the arc between 2 feature points. The reason why feature points are important is that we can assume that the arc that lies between 2 consecutive feature points does not belong to different circles. In other words, it is a complete circle without interference and we can use this arc to infer the whole circle.

Given 2 consecutive feature points $\boldsymbol{P1}$ and $\boldsymbol{P2}$, we first draw a line that connects the 2 points and denote it $\boldsymbol{L}$. In most case, the polygon that connects all the feature points is convex, which means that all the other vertexes lie on the same side of the edge connecting 2 given vertexes. So we can just randomly pick one remaining feature point, plug its position into the expression of L and then find the sign. Now we plug the position of all the points on the boundary to the expression. It is clear that the arc between 2 selected points will have a different sign and we remove all the points with the same sign.

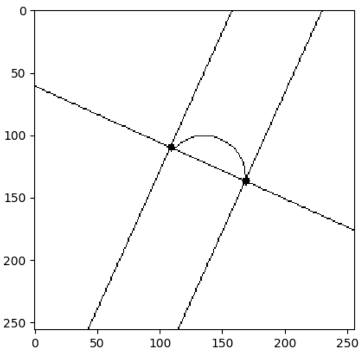


Figure 20: Arc Extraction

Sometimes this step will not remove all the other points that we are not interested in because some of the points that belong to other arc will also have a different. In that case, we draw 2 perpendicular line with respect to $\boldsymbol{L}$ that passes $\boldsymbol{P1}$ and $\boldsymbol{P2}$ and denote them $\boldsymbol{L1}$ and $\boldsymbol{L2}$. Similarly, we remove all the points that lie outside the area encompassed by $\boldsymbol{L1}$ and $\boldsymbol{L2}$ using the same sign deciding method. Although sometimes this will also remove some of the points on the interested arc, the remaining points are enough for us to decide the position and radius of the circle.

4.3.4 Finding Center and Radius

When we extract the arc that we are interested in, we can now find the center and radius of the circle that the arc belongs to.

To do this, we will first find the perpendicular bisector $\boldsymbol{PB}$ of $\boldsymbol{L}$. It is obvious that the center must lie on $\boldsymbol{PB}$. So for every point $\boldsymbol{S}$ on $\boldsymbol{PB}$, we calculate the distance between $\boldsymbol{S}$ and every point on the arc and store them in a array D_S. [8 pt] Also, we notice that if $\boldsymbol{S}$ is the center, then every point on the arc will have the same distance to $\boldsymbol{S}$, which means that the variance of D should be 0. As a result, we will just find the $\boldsymbol{S}$ with minimum variance of D_S. This point should be the center and the average of distance should be the radius.

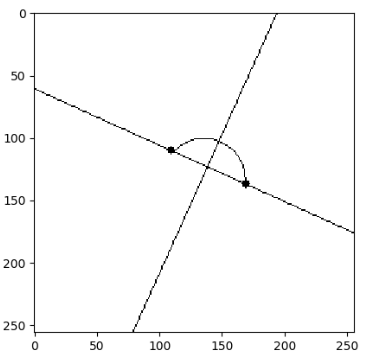


Figure 21: Perpendicular Bisector

$$todo \tag{11}$$

We do this step on every pair of the consecutive feature points and we will find all the candidates of the circles.

4.3.5 Merging and Removing Redundant Circles

In real practice, it is very common that the feature points we find do not really lie on the intersection of 2 circles due to the boundary noises. As a result, the algorithm may detect some redundant circles. So we must introduce some mechanism to merge the redundant circles. Besides, thanks to the unavoidable noises, some circles that do not belong to the bubbles might be detected, and we must come up with some ideas to remove them.

There are 2 ways to check and merge circles. The first one checks the distance between centers. If the centers of 2 or more circles lie too closely within some threshold, we will merge them to one circle by taking average of the centers and radii.

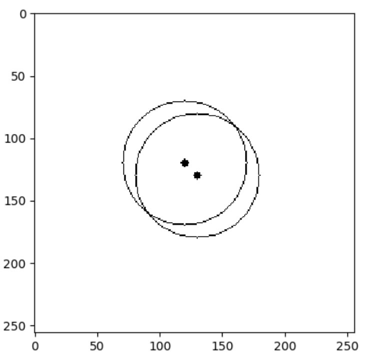


Figure 22: Closely Located Circles

The second way is to check the overlapping area. In real practice, chances are that one small circle lies within another large circle. Under this circumstance, the small circle should not be considered as a detected circle and must be remove. We can do this by checking the overlapping area of every pair of circles. If the overlapping ratio surpasses some threshold, we will remove the smaller circle.

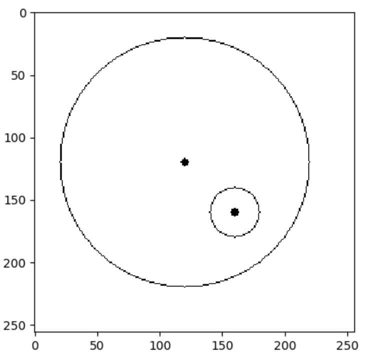


Figure 23: One Circle Within Another

For the second problem, there is also a way to tackle it. We will check the ration of the overlapping area between every detected circle and the original bubbles to the area that belongs only to the original bubbles or to the detected circle. We call it the ration of “overlapping “ to “differencing”. If the ratio is too small, then we can decide that the detected circle lies too much outside the original bubbles and should be removed.

4.3.6 Single Circle Detection

After so much algorithm introduced, we are finally able to detect the overlapping circles. However, there is still one thing left. Our algorithm does not work for single circle because our algorithm is based on finding feature points and according to our definition, a perfect circle does not have any feature points.

A common idea will be that if we do not detect any feature point, we will consider the image as one circle. The problem is that in real practice, there are many edge noises. So besides the feature points we desire, we will still find some other points that do not lie on the intersection of circles due to the unavoidable zigzags. Fortunately, these undesired points will not interfere our following detection thanks to the redundant circles merging mechanism. However, we have to find another way to decide whether the picture has just one circle.

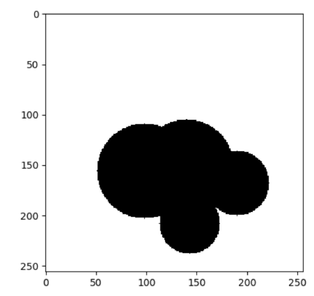
The idea is simple. We first find the leftmost, rightmost, uppermost and lowermost points in the foreground and use these points to decide the width and height of the foreground. If the difference of width and height surpasses some certain threshold, then the foreground cannot be one circle because the width and height of a circle are equal. If the difference is within the threshold, we then draw a circle using the four points and check the overlapping ratio just the same as we do in the previous steps. If the overlapping ratio surpasses certain threshold, we will recognize the bubble as one circle.

4.4 Performance Analysis

We tested the algorithm both on program-generated perfect overlapping circles and real overlapping bubble images.

4.4.1 Test on Program-generated Circles

A typical test image and its corresponding result are shown below.


(a) Program-generated Circles

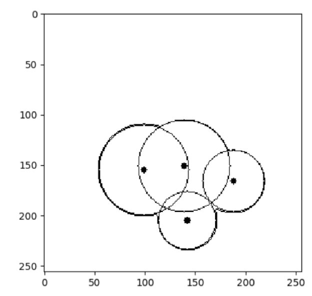
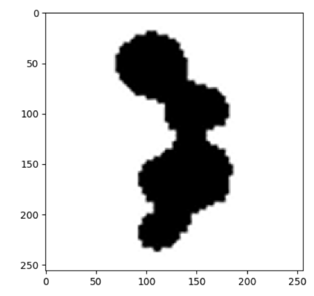

(b) Result

Figure 24: Test on Program-generated Circles

4.4.2 Test on Real Bubble Images

A typical test image and its corresponding result are shown below.


(a) Real Bubble Images

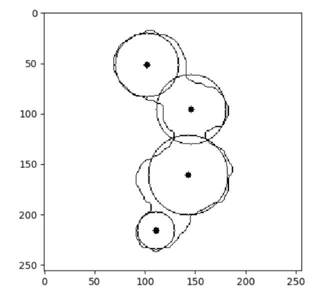

(b) Result

Figure 25: Test on Real Bubble Images

We tested the algorithm on 135 frames grabbed from real experiment bubble images and the accuracy reaches 89.6%.

5 Ultrasound Signal Recovery

Because the invisibility of ultrasonic wave, there is no direct method to see ultrasonic wave. In our experiment, we placed a laser generator in the one side of the water tank and we placed a light sensor in another side of the water tank. The laser generator would emit laser toward the water and the light sensor would generator electric signal whose frequency and amplitude are relative to light property. When there occurs waves in the water, the water refractive index will change which cause light intensity changed. So The light intensity signal measured by light sensor can reflect some characters of ultrasonic wave.

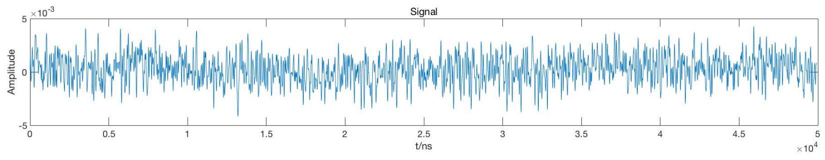


Figure 26: Sample Ultrasonic Waveform

However there is some drawback of this method. A major bottleneck encountered with such measurements is the ingress of external interference (usually of very high amplitude comparable to signal) directly affects the quality and reliability of the acquired signal data. Major external interferences encountered during on-site ultrasonic signal measurement and their sources are:

- Discrete spectral interferences (DSI) from radio transmissions and power line carrier communication systems.
- Noise caused by laser generators.
- Noise caused by impurity of water.

In addition to the above sources, other noise sources that can possibly exist in a ultrasonic measuring circuit are, random noise from components. In most cases, the amplitude of these external interferences so large that it is so hard to find the waveform of ultrasonic wave in the signal data, thereby reducing the credibility of the ultrasonic wave signal. Traditional method to solve this problem is to do average sampling. The most common average sampling number of this sort signal is 200. Only after this operation can we extract ultrasonic waveform from signal. With the advent of high-speed computers and fast A/D converters, digital signal measurement became a reality, and many digital noise suppression method evolved, such as, moving averages, FFT thresholding,digital filtering (infinite impulse response, IIR and finite impulse response, FIR), adaptive filtering. Most of the above mentioned methods mainly deal with the removal of sinusoidal interferences. It is now a commonly accepted fact that, removal of such narrow band interferences is not difficult. However, the main problem that continues to persist is that we have no information of when the signal occur and the center frequency of signal. The frequency of signal ranges from 2MHz to 15Mhz, which is a very large range.

5.1 Drawback of Fourier Transform and Short-time Fourier Transform

Fourier Transform is a integral decompose method to decompose signal into sin signals and cos signals. Through it we are able to know the frequency band of signal, but we can't not know when this signal occur and how long that signal last. So we lose the information of the time. To conquer the drawback of Fourier Transform, short-time Fourier Transform is proposed. The idea of it is to divide signal into many shorter segments by window function. And then compute Fourier transform separately on each shorter signal segment. By changing the size of time period we can plot the changing spectra as a function of time. However, Short-time Fourier Transform also has its pitfalls. One of the pitfalls of the STFT is that it has a fixed resolution. The width of the windowing function relates to how the signal is represented—it determines whether there is good frequency resolution (frequency components close together can be separated) or good time resolution (the time at which frequencies change). A wide window gives better frequency resolution but poor time resolution. A narrower window gives good time resolution but poor frequency resolution. These are called narrowband and wideband transforms, respectively.

5.2 Wavelet Transform

5.2.1 Brief Introduction To Wavelet Transform

A wavelet is a small wave, which has its energy concen- trated in time, and a tool meant for analysis of transients and non-stationary or time varying signals. Just as Fourier analysis consists of decomposing a signal into sine waves of various frequencies, similarly, wavelet analysis is the breaking up of a signal into shifted and scaled version of mother wavelet.Continuous wavelet transform (CWT) of a signal $x(t) \subseteq L^2(R)$ is defined as

$$CWT^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^{*}\frac{t-\tau}{s}dt$$

where, function $\phi(t)$ is the mother wavelet. It is ,a prototype for generating the other window functions, which are dilated or compressed and shifted versions of mother wavelet. ϕ the shift operator (translation), s is the scaling function and $*$ stands for complex conjugation. Wavelet transform maps a time-domain signal into a two dimensional array of coefficients, thus localizing the signal in both time and frequency domain simultaneously, whereas, Fourier transform can only give the frequency information. However, CWT is computationally expensive and also generates a lot of redundant data. To circumvent these drawbacks, an effective implementation applicable to discrete signals, called discrete wavelet transform (DWT) was formulated using suitable lowpass and high- pass filters, that satisfy certain mathematical constraints. An elegant procedure called the Multi-resolution Signal Decomposition (MSD) technique is implemented through this method, which is the primary reason. for the widespread use of wavelets.

5.2.2 Discrete Wavelet Transform

In Discrete wavelet transform, a time-scale representation of a digital signal is obtained using digital filtering technique. Filters of different cutoff frequencies are used to analyze the signal at different scales. The time-domain signal is passed through a series of highpass filters and down-sampled by two to analyze the high frequencies (referred to as detail components), and it is passed through a series of lowpass filters followed by down-sampling by two, to analyze the low frequencies (called approximate components). These highpass and lowpass filters constitute 'quadrature mirror filters', and are exactly half-band filters, thus enabling a perfect error-free reconstruction of the signal. For reconstruction, the above procedure is fol- lowed in reverse order i.e. the signals at every level are up-sampled by two and passed through a set of synthesis filters (synthesis filters are derived from analysis filters). Thus, for an M level decomposition-reconstruction, the input signal can perfectly be recovered by adding the reconstructed time- domain approximate component at level M and all the reconstructed time-domain detailed components from level 1 to M i.e.

$$ReconstructedSignal = (Approximatecomponent)_M + \sum_{j=1}^{M} (Detailcomponent)_j ...$$

The MSD method is inherently error-free, and this hap- pens to be a big advantage, since the process by itself does not introduce any additional error.

5.3 Proposed Technique

The conventional wavelet denoising method involves calculation of the DWT coefficients for a given signal and then passing the DWT through a threshold (fixed a priori), and followed by reconstruction of the signal by taking the inverse wavelet transform of the modified DWT coefficients. This method is known as soft or hard thresholding. In the present problem, the amplitude of noise of the noise is almost the same as signal. So soft thresholding and hard thresholding will not pursued further in this paper. The proposed technique is basically an off-line technique and has the advantages of low processing time and possibility of better reconstruction, because the suppression of noise interferences is done in a joint time-frequency domain. This type of denoising consists of two phases. In the first phase, input signal is decomposed into approximate and detail components up to a desired number of levels with the help of multiresolution analysis. This is done by first choosing a mother wavelet according to the signal characteristics. Once, the mother wavelet is chosen, decomposition-reconstruction up to the required number of levels is carried out by, scaling and dilating the mother wavelet. The numher of decomposition-reconstruction levels can he chosen either by trial and error method or, as in this work, was chosen from the information of sampling frequency. This is because the frequency band of each decomposed component follows a dyadic rule, starting from the Nyquist frequency. Once the approximate and detail reconstructed time-domain components are computed, the second phase starts. This phase involves identifying those components that correspond to all the ultrasonic signals, either by finding max energy point or by the knowledge of frequency hands to which the PD pulses belong (this is known from the bandwidth of the PD detector). Finally, the denoised signal is obtained by discarding all the identified components from the summation process. In summary, the proposed wavelet method involves the following three steps:

1. Using a mother wavelet, the input signal is decom- posed into a $pre - set$ (M) number of levels, and this yields DWT coefficients.

2. The earlier methods worked with these DWTs. But, in the proposed method, the DWTs are reconstructed to yield M levels of time-domain sequences (each of length equal to original signal length). Each of these M levels, correspond to a band of non-overlapping frequencies, and adding all M levels yields the original signal. Thus, a joint time-frequency representation of the input signal is obtained.

3. Denoising in the present method essentially consists of a visual inspection of the M reconstructed time-domain components (or levels), and identifying those that lie within the pass-band of the ultrasonic signal. In some special cases, one or two extra bands on either side of the band-pass may have to be chosen. Depending on sampling frequency and levels of decomposition, it is easy to locate the bands that are to be retained. The rest of the time-domain components are unwanted and discarded from the summation process.

5.4 Issues In Wavelet Method

During the implementation of the MSD approach, choice of some parameters connected with MSD method has to be made. This is inherent to the method, and concerns issues like choice of mother wavelet, number of levels of decomposition, etc. They are briefly addressed below:

1. Selection of mother wavelet. This is a core question often asked, but a generalized answer seems to be still elusive even to signal processing experts. Also, no direct answers are available in literature. With this being reality, its choice in the present work was based on a trial-and- error and guided by hints published in literature on simi- lar type of work. Being well aware of this issue, the au- thors have examined many wavelets and found Daubechies' wavelet most suitable. Further, many practi- cal examples have been included to demonstrate that the wavelet selected was good enough for the task. A detailed comparative study of all wavelets was not exclusively re- ported, as this was not within the identified scope of the paper.

2. Selection of number of levels for decomposition- reconstruction. The goal was 'to have sufficient resolution in frequency hands at lower frequencies, as low frequency pulsive interferences were also to be eliminated. If lesser levels were used, then PD signals and the low frequency interferences would be clubbed together, and hence diffi- cult to segregate. Keeping this in mind, ten decomposition-reconstruction levels were chosen and found to be sufficient in most of the cases.

5.5 Digital Simulation Setup

5.5.1 Basis For Comparison Of Technique

During the process of denoising, in addition to removal of noise components, the signal component whose frequency is very close to the interfering frequency ranges also gets removed (although, to a lesser extent). Thus, the signal component to be recovered suffers from both attenuation and distortion. These two features may be quantified by defining the peak amplitude and the correlation coefficient. An acceptable noise suppression method should reject or suppress all the interferences and noise with minimum attenuation and distortion of the ultrasonic pulse. In order to compare the performance of various methods, the following indices have been considered:

Signal To Noise Ratio

The signal to noise ratio (SNR), illustrates the effectiveness of denoising operation. It is defined as

$$SNR = 10 * log \frac{\sum_{i=1^N} Y^2(i)}{\sum_{i=1}^{n} (X(i) - Y(i))^2}]$$

where,$X(i)$ is the original reference signal, $Y(i)$ is the denoised signal and N is the number of samples. A positive value of SNR implies a greater power of the signal as compared to the noise and a negative value of SNR implies a greater power of noise as compared to the signal. As in practical records, there is no such reference signal $X(i)$ to compute SNR, only the extent of noise suppressed can he estimated. The normalized noise reduction was computed as:

$$reductioninnoiselevel(dB) = 10 * log \sum_{i=1^N} \frac{1}{N} (Z(i) - Y(i))^2$$

where, $Z(i)$ is the noisy signal acquired, $Y(i)$ is the denoised signal and N is the number of samples.

Reduction In Ultrasonic Amplitude

The percentage reduction in amplitude of the denoised signal with respect to the reference signal is defined as

$$reduction = \frac{X - Y}{X} * 100$$

where, X is the peak amplitude (positive going peak in this case) of the original(reference) signal and Y is the amplitude of the denoised signal recovered.

5.6 Results and Discussion

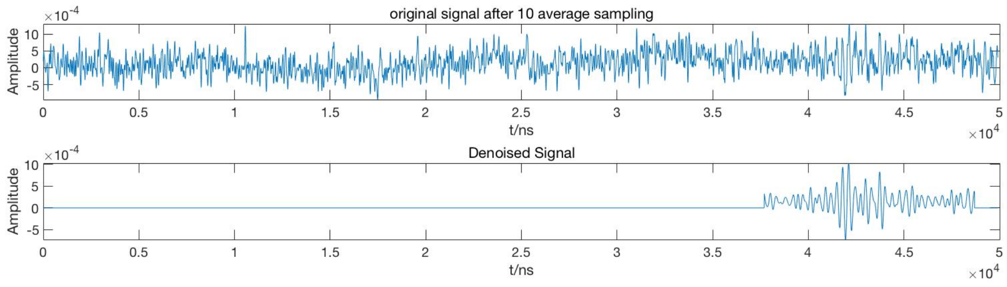


Figure 27: Result of processing Signal

Accuracy	SNR	Reduction in Amplitude
69%	cell5	cell6

From the table above, we can know that this method can't identify the ultrasonic wave from noise in some cases. Because in these case the energy of noise is larger than the energy of signal. Apart from that, the data o SNR and reduction in amplitude reveal that our method is effective and it can help us to find signal better.

6 Acknowledgments

First and foremost, We would like to thank Professor L. Jay Guo for inviting us to participate in the inter-university research training program and advising our team, without whose ideas and guidance our work wouldn't have been made possible. We also thank Ph.D candidate Qiaochu Li for supporting our project all the way through and provide valuable input. It is his proposed "end need" that gives the incentive to our project. Finally, we acknowledge the financial and infrastructural support from University of Michigan and EECS department for hosting our visit.

Bibliography

[1] M. Piccardi, "Background subtraction techniques: a review," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 10 2004.

[2] "Image source: `https://www.codeproject.com/Articles/142859/Extended-GMM-for-Background-Subtraction-on-GPU`,"

[3] "Image source: `https://cloud.tencent.com/developer/article/1011661`,"

[4] L. W. Qingxiong Yang and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," *Computer Vision and Pattern Recognition (CVPR)*, 03 2010.

[5] A. A. Muhammad Nawaz, John Cosmas, "Foreground detection using background subtraction with histogram," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2013.

[6] R. S. Friedman, N., "Image segmentation in video sequences: a probabilistic approach.," *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, 1997.

[7] G. W. Stauffer, C., "Adaptive background mixture models for real-time tracking," *Proc. of the Conf. on Computer Vision and Pattern Recognition*, 1999.

[8] S. J. Power, P.W., "Understanding background mixture models for foreground segmentation," *Proc. of the Image and Vision Computing New Zealand*, 2002.

[9] E. J.-O. Hayman, E., "Statistical background subtraction for a mobile observer," *Proc. of the Internat. Conf. on Computer Vision*, 2003.

[10] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 03, pp. 773–780, 2006.

[11] "Image source: `https://blog.csdn.net/tMb8Z9Vdm66wH68VX1/article/details/79983422`,"

[12] H. D.-D. L. Elgammal, A., "Non-parametric background model for background subtraction," *Proc. of the European Conf. of Computer Vision*, 2000.

[13] "Reference: `https://docs.opencv.org/3.1.0/d9/d61/tutorial_py_morphological_ops.html`,"

[14] H. Li, M. A. Lavin, and R. J. Le Master, "Fast hough transform: A hierarchical approach," *Computer Vision, Graphics, and Image Processing*, vol. 36, no. 2-3, pp. 139–161, 1986.

[15] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.