CRAB COUNTER

Wei Liu and Ke Chen

Dec.15, 2007

Boston University

Department of Electrical and Computer Engineering

Technical report No. ECE-2007-04

BOSTON
UNIVERSITY

CRAB COUNTER

Wei Liu and Ke Chen

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec. 15, 2007

Technical report No. ECE-2007-04

Summary

This work was part of the course EC520. The objective of this project is to develop an algorithm for the detection and enumeration of crabs on beach. The basic method is background modeling and background subtraction. Based on several previous frames, the background of current frame can be estimated using median model or non-parametric model. Morphological operation is applied to foreground images and the number of crabs can be obtained. The method is tested on an actual video of crabs on beach. Comparison of different algorithms is made.

Contents

List of figures and tables

1 Introduction

The remarkable shrinking of network video cameras in both size and cost has led to a serious effort of using networks of such cameras for wildlife monitoring. A NSF-funded project is building such a video camera network and using it for active monitoring of Atlantic shore wildlife. Based on the video data of crabs moving on the beach, our task is to develop a method to detect the presence of crabs and then count their number.

2 Literature Review

2.1 General idea for moving objects detection

1. Background

Background subtraction is a commonly used class of techniques for detecting all foreground objects in a scene for application such as surveillance. It involves comparing an observed image with an estimate of the image contained no objects of interest, which is the so called “background”. A simple technique of “comparing” is to subtract the observed image from the background, and then threshold the result to generate the objects of interest.

2. Non-static background

As a matter of fact, the background might not be fixed but adapts to:

1). Scene’s illumination changes, e.g., gradual change from morning to evening or sudden change due to clouds;

2). Motion changes because of camera oscillation or high-frequencies background objects (such as tree branches and sea waves) movement, which cause the pixel intensity values to vary significantly;

3). Changes in background geometry

4). Etc.

2.2. Widely used techniques for estimating the scene's static background

1. Basic methods

1) Frame difference:

$$\text{difference image} = frame_{i+1} - frame_i$$

Then threshold the difference image to detect the foreground objects:

$$|\text{difference image}| > \text{threshold}$$

In this straightforward method, the background is just the previous frame and the result is very sensitive the threshold. It only works well in a particular condition.

2) Average or median of the previous N frames:

Usually, during N consecutive frames, a fixed pixel is background most of the time, and is foreground only when the objects of interest move across. So the background can be estimated by average of the N intensity values of the pixel, or median of that:

$$\text{background image} = NINT\left\{\frac{1}{N}\sum_{i=1}^{N} frame_i\right\} \qquad \text{NINT: Nearest integer}$$

or

$$\text{background image} = \text{median } \{\text{N frames}\}$$

3) Running average or running median:

$$background_{i+1} = \alpha * frame_i + (1-\alpha) * background_i$$

where $background_i$ is the background image obtained by average model or median model. Each pixel's location is based on the pixel's recent history. The recent frames have higher weight, e.g., α =0.05, typically.

4) Running average or running median with selectivity:

$$background_{i+1} = \begin{cases} \alpha * frame_i + (1-\alpha) * background_i & (if \quad frame_i \quad is \quad background) \\ background_i & (if \quad frame_i \quad is \quad foreground) \end{cases}$$

The feedback from the classification of a pixel as foreground or background prevents the background model to be polluted by pixels logically considered as foreground.

2. Density estimation methods

1) Single Gaussian:

If we monitor the intensity value of a pixel over time in a completely static scene (i.e., without background motion), then the pixel intensity can be reasonably modeled by Gaussian probability density function (pdf.): N(μ ; σ^2), given the image noise over time can be modeled by N(0; σ^2). That also explains why we can use average image of N frames as background.

2) Running Gaussian average

Update the mean value (μ) and variance (σ^2) of Gaussian pdf. of a fixed pixel:

$$\mu_{i+1} = \alpha * frame_i + (1-\alpha) * \mu_i$$

$$\sigma_{i+1}{}^2 = \alpha * (frame_i - \mu_i)^2 + (1-\alpha) * \sigma_i{}^2$$

where $frame_i$ is the intensity value of this pixel in the i^{th} frame.

3) Mixture of Gaussians

Estimate the pdf. of the intensity value of a pixel as:

$$P(u) = \sum_{i=1}^{K} \omega_i N(\mu_i; \sigma_i{}^2)(u)$$

Pre-define K=3~5. All the μ_i, $\sigma_i{}^2$ are updated at every new frame. The mixture of Gaussians actually models both the foreground and the background. All distributions are ranked according to their ω_i / σ_i and the first ones chosen as "background".

4) Kernel density estimators

It is a generalization of the mixture of Gaussian model. We will discuss it in detail in section 3.

3. Suppression of false detection

False detection of foreground might be resulted from random noise, non-static background or camera oscillation. There are 3 commonly used methods:

1) Low-pass filter or median filter: They can remove random noise, but often fail when some moving background objects are detected as foreground.

2) Morphological operation: Erosion, Closing and etc. to remove random noise and pixels on the boundaries of objects in the images.

3) Another way which is based on Kernel density model (or Mixture of Gaussians model), so we will discuss it in depth in Section 3.

3 Problem Statement

Our task is to implement a method to detect and count crabs, in order to find the trend how the number of crabs changes over time.

In monitoring application, if we want to count the crabs in the (N+1)th frame, its background is estimated based on previous N frames. If we want to count the crabs in the (N+k) th frame (k is a small integer, e.g., k=5), we need to update the background of the (N+1)th frame.

However, in this project, we focus on developing a method to detect and count crabs in one frame, instead of counting crabs in all the frames of this video. So, we didn't implement the algorithm for updating background.

3.1 Challenges in Detection

The detection of crabs is based on their movement, not on their shape, size, color, etc. If a crab is at rest, it cannot be detected. These are many difficulties occurred in such detection:

1. Crabs which are difficult to separate

1) The video of crabs on the beach doesn't have high enough spatial resolution, while the crabs are very small compared to the scene size, and easy to be covered by noise. Thus, this problem is more difficult than detection of relatively large objects such as cars.

2) There are nearly one hundred crabs on the beach and some of them crowd in a small area, even overlap between each other. Moreover, those crabs are not of the same size and have similar color to the sand.

2. Non-static Background

The most serious problem is water ripple which is non-static background. It will be detected as foreground since the detection is based on movement. So it will cause a lot of false detection and can't be significantly removed by simple methods such as low-pass filtering. As a result, the crabs in the water are very difficult to detect. In addition, water area has similar color and luminance to sand, so it’s not very straightforward to separate water from the scene.

3. Camera oscillation also introduces noise in foreground image.

3.2 Basic Assumptions

As discussed above, it’s not easy to detect and count all the crabs. In order to simplify the problem, we make the following assumptions:

1. The crabs that don’t move during the frames will be used to build background and are not counted in. Our task is only to detect the moving crabs.

2. Crabs that crowd in a small area or overlapped each other, will be considered as one crab.
3. There is no camera oscillation.

3.3 Method Description

3.3.1 Background Modeling

For this project, the background is not static because of water ripple. So, some methods such as average model and single Gaussian model are not good estimation. Median model is acceptable but not good enough. Kernel density estimator is a better model, because it takes statistic properties of the image sequence into consideration.

1. The structure of probability density function

Let $\{u_1, u_2, ..., u_N\}$ be a recent set of samples of intensity values for a pixel. Given this sample set, the pdf. of this pixel can be non-parametrically estimated using the kernel estimator K( ) as:

$$P(u) = \frac{1}{N}\sum_{i=1}^{N} K(u - u_i)$$

Often, we choose our kernel estimator function K( ) to be Gaussian: $N(0;\sigma^2)$, where σ represents the kernel function bandwidth, i.e., the standard deviation. Then the density can be estimated as:

$$P(u) = \frac{1}{N}\sum_{i=1}^{N} N(0;\sigma^2)(u - u_i) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(u-u_i)^2}{2\sigma^2}]$$

Note:

For a fixed pixel, only based on its intensity value in the k^{th} frame (assuming it's u_k), the probability density that its intensity value is u is $N(u_k;\sigma^2)(u)$. So if we observe N recent frames, (where N is a relatively large number, e.g. 50), based on the N intensity values of this pixel, we can estimate the pdf. of its intensity

value as the summation of the N Gaussian pdf. as above. This summation finally approximately equals to the summation of M Gaussian pdf., (where M is a relatively small number, e.g. 3), each of which models one object in the scene, such as sand, water, crabs, etc.

2. Normalization of pdf.

According to the definition of pdf., its integral over the field of definition must equal to 1, so normalization of the obtained pdf. is necessary. However, the obtained "pdf." is actually discrete and is defined on [0,255], because the possible intensity value of a pixel is between 0 and 255. For simplicity, we just normalize it by multiplying 1/N. In fact, this won't affect the accuracy of the result, which is discussed below.

3. Use the pdf. to decide if this position is background in a new frame

Based on N frames, we find the pdf. for every pixel's intensity value. In an application of surveillance, for any pixel in the scene, it is background most of the time and foreground only when an event "happens" at this position, e.g., a crab moves across. So the value with higher probability density implies that it is probably background.

Using the obtained pdf. for a pixel, we examine the same position in an observed frame. Assuming the intensity value of this pixel is u_t, if $P(u_t)$ <TH, i.e., u_t has a relatively lower probability density, this pixel is considered as foreground. Otherwise, it's considered as background. TH is a global threshold over the whole image. With smaller TH, more pixels are considered as background, so we have less false positives but might lose some true detection. So we can adjust TH to achieve a desired percentage of false positives.

Because that we compare the pdf. with TH, but don't use it to compute probability, the normalization of pdf. is not necessary in this sense. As a result, how we implement the normalization won't affect the accuracy of result.

4. Estimating σ^2 in the pdf. of a particular pixel

We use the same σ^2 for each Gaussian pdf. in the summation, where σ reflects the small deviation of the pixel's intensity value resulted from blurring in the scene (i.e., local variance), but not from displacement of different objects. Actually, the significant variance due to objects moving has been taken into consideration. That's why we use the summation of N Gaussian instead of a single Gaussian.

This local variance of each pixel changes over time. σ can be estimated as:

$$\sigma = \frac{m}{0.68\sqrt{2}}$$

where $m=\text{median}\{|u_{i+1}-u_i|\}_{i=1}^{N-1}$, $(u_{i+1}-u_i)$ is each consecutive pair among N frames.

As assumed, the intensity value of a pixel can be modeled by $N(\mu;\sigma^2)$, if no background objects move across it. Thus, the difference of intensity values of this pixel between two consecutive frames can be modeled by $N(0;2\sigma^2)$, i.e., $(u_{i+1}-u_i)\sim N(0;2\sigma^2)$.

3.3.2 Suppression of False Detection

The main idea of suppression of false detection is to find out which of those pixels detected as foreground are in fact background. If part of the background (e.g., a tree branch) moves to occupy a new pixel (e.g., [i,j]), but in the model, that pixel was not considered as this part of background (i.e., $P_{ij}(u[i,j]) < TH$, where $P_{ij}(\bullet)$ is the pdf. of the pixel [i,j]), then it will be detected as foreground. However, this moved part of background has a high probability that at its original pixel (e.g., [k,l]), it was considered to be background ($P_{kl}(u[i,j]) > TH$,where $P_{kl}(\bullet)$ is the pdf.

of the pixel [k,l]). Assuming that only small movement occurs during consecutive frames (i.e., [i,j] is very close to [k,l]), the pixel which is detected as foreground (i.e., [i,j]) has a high probability that it would be considered as background, if we use its neighbors' pdf.:

The detected foreground pixel [i,j] is background if:

$$P_N(u[i,j]) = \max_{[k,l]\in N_{ij}} \{P_{kl}(u[i,j])\} > TH_1$$

Note:

$P_{kl}(u[i,j])$ gives the probability density that the intensity value of pixel [i,j] at its possible original position [k,l] equals to .u[i,j]. Larger $P_{kl}(u[i,j])$ implies higher probability that pixel [i,j] at its possible original position [k,l] is considered as background. Thus, to decide if pixel [i,j] at its possible original position [k,l] is actually considered as background, we only need to compare the maximum probability with TH_1.

By thresholding $P_N(u[i,j])$, we can eliminate many false detections due to small motions in the background. Unfortunately, we also eliminate some true detections, because some true detected pixels might accidentally have similar intensity value to some nearby pixel which are considered as background. This happens more often on gray level images.

To avoid losing such true detections, we add the constraint that the whole detected foreground object must have moved from a nearby location, and not only some of its pixels. The implementation of this idea has quite high computational complexity, and we will try it to improve the result later on.

3.3.3 Real-time Processing

The application of surveillance requires real time processing. Once an event is detected (e.g., crabs moving), corresponding action is taken (e.g., counting the number of crabs). However, the computational complexity on video sequence is always high. Thus, optimization of algorithm is quite necessary, and we will discuss it in detail in section 4.

3.3.4 Enumeration of the Crabs

With the processed foreground image (binary image, background pixels are black and foreground pixels are white), we count the crabs by using morphological operation.

1. Label the connected areas in region $\mathbf{D}$. Assume that:

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

As commonly used definition of neighborhood in image, there are two kinds of neighborhood: 4-neighborhood and 8-neighborhood. We decide if two pixels are connected or not by checking if they are neighbors. Examples as below:

4-neighbor label: $\mathbf{L} = \begin{bmatrix} 1 & 1 & 0 & 2 & 2 & 0 \\ 1 & 1 & 0 & 2 & 0 & 4 \\ 1 & 1 & 0 & 0 & 0 & 4 \\ 1 & 1 & 0 & 0 & 3 & 0 \end{bmatrix}$

8-neighbor label: $\mathbf{L} = \begin{bmatrix} 1 & 1 & 0 & 2 & 2 & 0 \\ 1 & 1 & 0 & 2 & 0 & 2 \\ 1 & 1 & 0 & 0 & 0 & 2 \\ 1 & 1 & 0 & 0 & 2 & 0 \end{bmatrix}$

Thus, there are 4 “crabs” in region $\mathbf{D}$ if we use 4-neighbor label, and 2 “crabs” in region $\mathbf{D}$ if we use 8-neighbor label.

2. Threshold the labeled image

After labeling, we threshold the number of pixels in the same connected area. If it not smaller than some threshold, this area is considered as an object, i.e., a crab; otherwise, this area is considered as noise and its label is removed, i.e., forced to zero. Then we re-label this region, and the largest label number is the number of crabs in the region.

4 Implementation

4.1 Detection

Detect the moving crabs, including estimating background, subtracting background from observed image, suppressing false detection.

We implement it in two ways: median model and Kernel estimation, where the latter one gives better result and the former one is for comparison.

1. Median Model:

Algorithm

- Build up background model M based on pervious N frames:
  1) Take N previous frames
  2) For coordinates at [i, j],
     a. Look for median m[i,j] in set $\{u_1[i,j], u_2[i,j], ..., u_n[i,j]\}$
     b. Define m[i, j] as value of the background model $M_{i,j}$
- Detect crabs:

Given the frame f_t, let Nt denotes the number of detected crabs in f_t:

1) Subtract the background M from the observed frame f_t and take absolute value to the difference image

2) Convert the grayscale image to a binary image by thresholding: difference greater than threshold will be valued 1 as foreground; otherwise, valued 0 as background.

3) Reduce noise using median filter

2. Mixture of Gaussian Model:

Algorithm

- Build a Lookup table S($256\times256\times256$ matrix)

XData, valued from 0 to 255, represents all possible intensity values of a pixel;

ZData, also valued from 0 to 255, represents all possible integer values for standard deviation;

YData is sampled probability on Gaussian density function.

For a pixel at coordinates [i,j] with value u and standard deviation σ, the probability for intensity value of this pixel to be v is given by $S(u,v,\sigma)$, namely:

$$S(u,v,\sigma)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\frac{(u-v)^2}{\sigma^2}}$$

- Build kernel density estimator $\mathbf{E}_{i,j}$

1). Take N consecutive frames

2). For coordinates at [i,j]

a. Compute $\Delta u_t[i,j]=|u_t[i,j]-u_{t+1}[i,j]| \quad t=1,2,...,n-1$

b. m=median $\{\Delta u[i,j]\}$ and define standard deviation $\sigma_{i,j}$ as: $\sigma_{i,j}=\frac{m}{0.68\sqrt{2}}$

c. Look up in table S for intensity probability distribution $\{P(U=v)\,|\,v\in[0,255]\}$ using $u_t[i,j]$ and $\sigma_{i,j}$, and summate over t

d. Normalize the summation and record into Estimator $\mathbf{E}_{i,j}$

- Object Detection Step

1) Take frame f_t

2) For coordinates at [i, j] with value $u_t[i,j]$

a. Look for $P(U = u_t[i, j])$ in estimator $\mathbf{E}_{i,j}$

b. Compare to threshold T_m, let $D_{i.j} = 1$ if $P(U = u_t[i, j]) < T_m$, or $D_{i.j} = 0$ if $P(U = u_c[i, j]) \geq T_m$, where $\mathbf{D}$ is the detected foreground image.

3) Reduce noise using median filter and morphological operations.

3. Improvement

The look-up table can be improved further more to get faster and more accurate computation. Also, we can stop the summation when we already have: $\sum_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(u-u_i)^2}{2\sigma^2}] > T_m N$, for $k \leq N$. However, due to limited time, we haven't finished the implementation of these two improvements and leave these as some future work.

4.2 Enumeration of the Crabs

Algorithm

1) Apply 4-neighbor label (or 8-neighbor label) in the processed image;

Let L denote the largest label used in the image, which equals to the number of detected objects. Initialize the total number of detected crabs Nt=0:

For $l = 1, 2, ...L$:

2) Find and count elements in D labeled l;

3) Threshold(T_c) the total number of pixels in a connected area obtained in step (1) by labeling, assuming this number is n_l:

If $n_l \geq T_c$, define object labeled l as crab, and increase Nt by 1;

If $n_l < T_c$, remove this object by forcing the value of those pixels in this object to 0, i.e., background;

4) Re-label, and then the largest label is the total number of crabs.

5 Experimental results

5.1 Result of Median Model

Original image to be observed: #351 frame from video crab1.avi

Fig. 1. Observed image

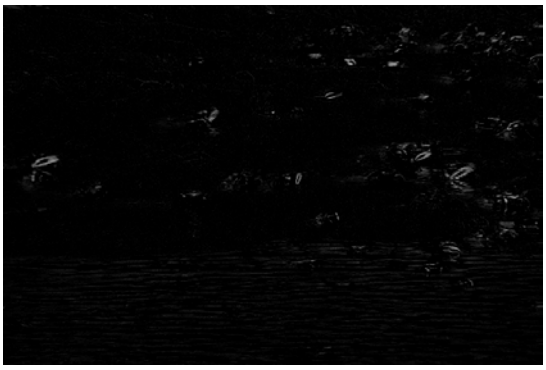
1. How the three parameters affect the results:

-- N: the number of frames used to build up background

-- T_G: the threshold used to converting gray level image to binary image

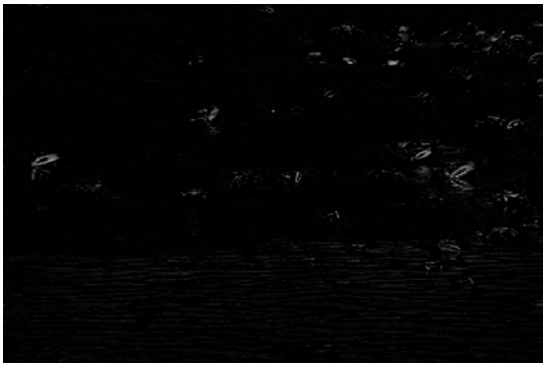
-- Size and shape of median filter mask

(a) N=50



(b) Detected foreground using (a)

(c) N=100



(d) Detected foreground using (c)

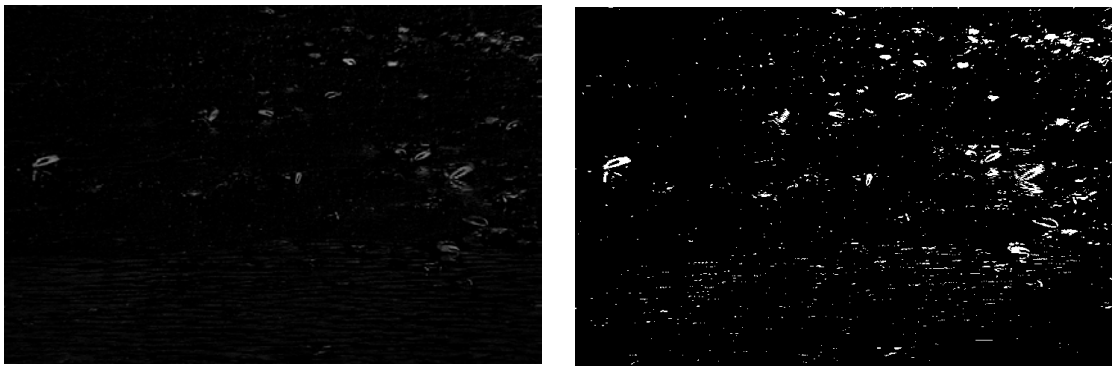
(e) N=300

(f) Detected foreground using (e)

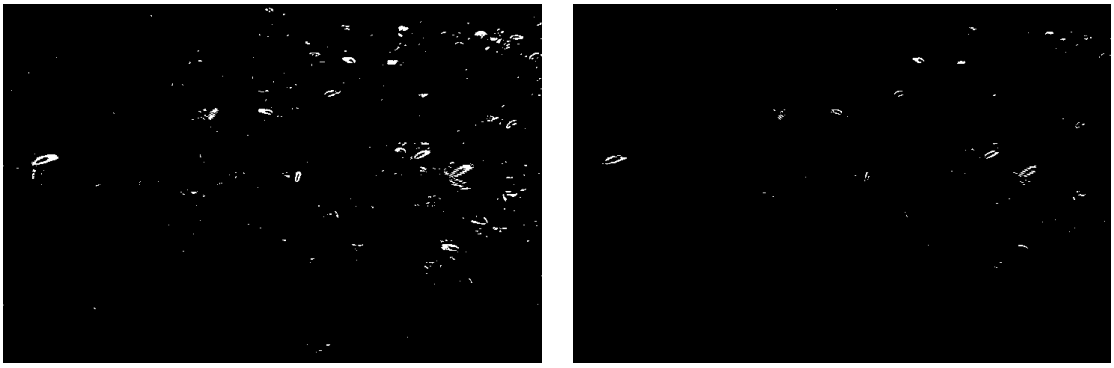
Fig. 2. Effect of changing N:
the obtained median models and detected foreground.

It's more likely to find medians that are sufficiently representative of the background with a larger N (number of frames based on which median model was built), and thus build a better estimation to detect the foreground.

However, we can see that the detected crabs are only their claws, because the body of crab and the beach has quite similar color and thus, in similar gray level. In addition, the detected crabs and the water ripple noise are in similar gray level, which makes it very difficult to remove noise without losing crabs. So, median model is not a very good model for this application.



(a) Fig 2(f) (b) Thresholding (a), T_G = 0.1

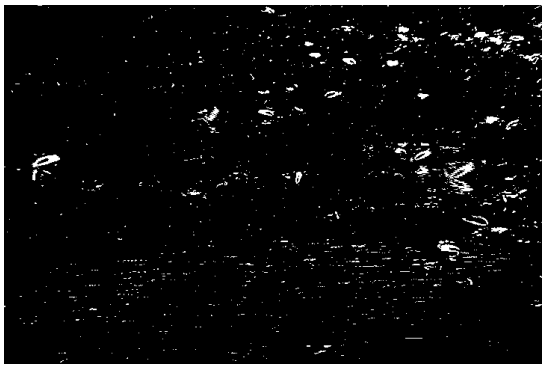


(c) Thresholding (a), T_G = 0.2 (d) Thresholding (a), T_G = 0.4

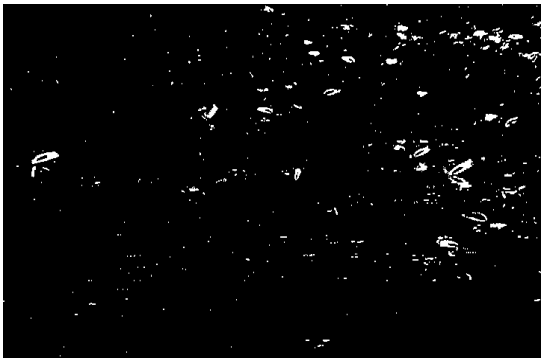
Fig. 3. Effect of changing graylevel threshold T_G

There is a tradeoff between crab detection and noise cancellation. Smaller TG would reserve more information about crabs, but leave noises in the image; larger TG would better suppress noises, but lose more information about crabs at the same time.

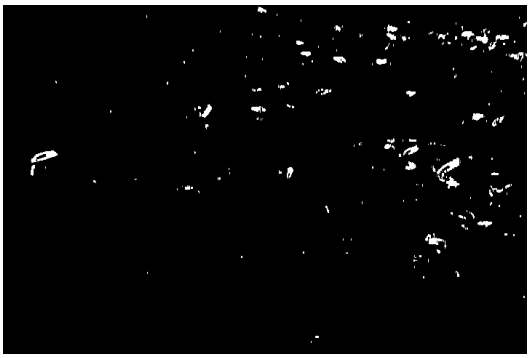
(a) Fig. 1



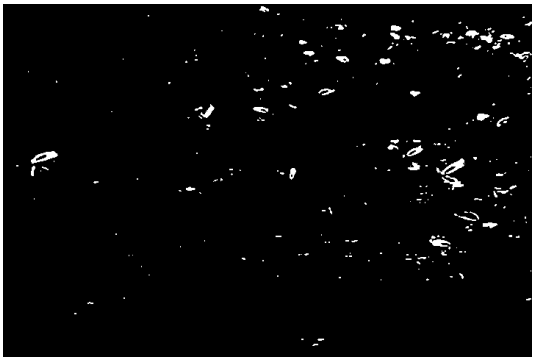
(b) Fig. 3(b)



(c) Vertical mask [3 1]



(d) Vertical mask [5 1]



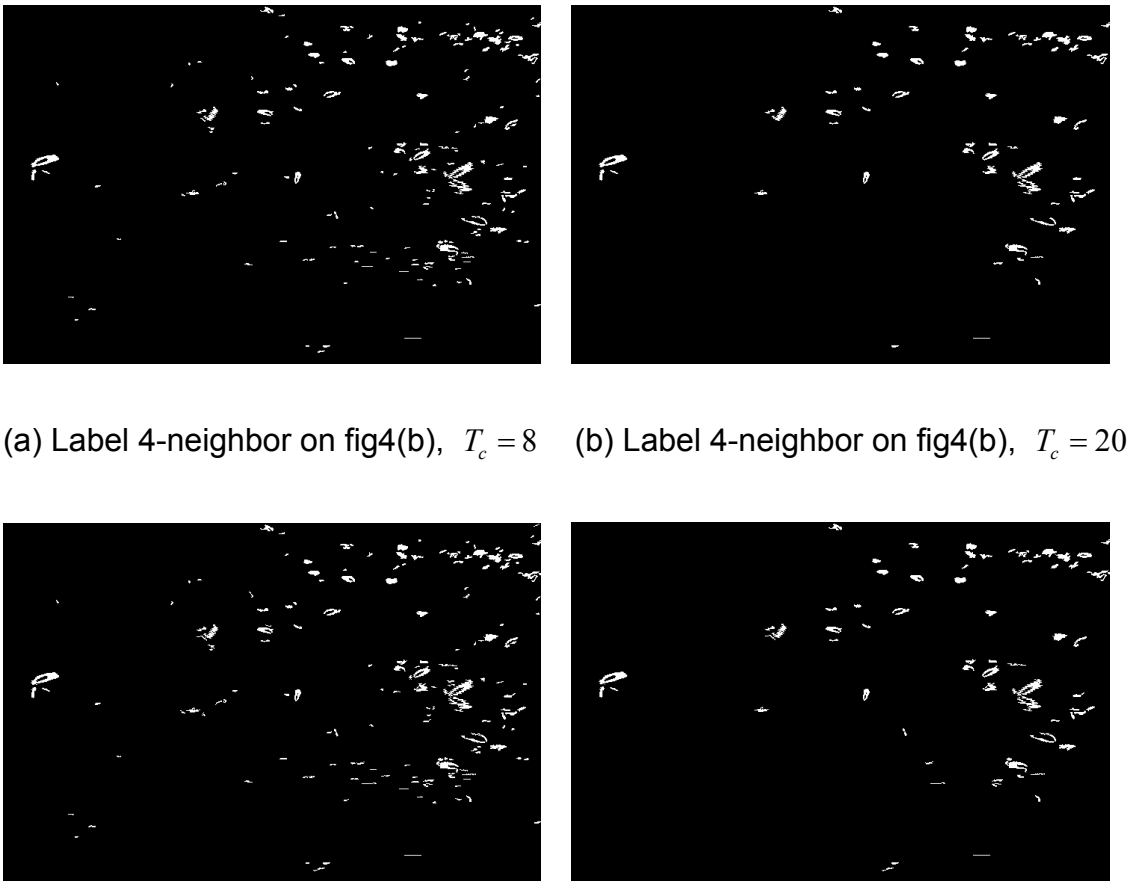
(e) Vertical mask [7 1]

(f) Square mask [3 3]

Fig. 4. Noise cancellation using different median filter masks

Results would largely depend on the shape and size of the masks: vertical mask turns to be more effective in getting rid of noises generated by water ripples, which are mainly horizontal.

2. Enumerate the number of crabs and how the two parameters affect the results:
-- 4-neighbor label or 8-neighbor label
-- T_c: threshold the number of pixels in a labeled connected area



(a) Label 4-neighbor on fig4(b), $T_c = 8$ (b) Label 4-neighbor on fig4(b), $T_c = 20$

(c) Label 8-neighbor on fig4(b), $T_c = 8$ (d) Label 8-neighbor on fig4(b), $T_c = 20$

Fig. 5. Crab detection using different neighborhood definition and T_c

Comparing (a) & (b), or (c) & (d), we can see that noises, appeared as small white dots in the detected foreground, can be removed since they include small number of pixels.

4-neighbor labeling and 8-neighbor labeling make not much difference to the result. In fact, with the same T_c, 8-neighbor labeling can leads to more false positive and less miss detection. Compare (a) & (b) or (c) & (d), we can see that smaller T_c leads to more false positive and less miss detection. In other words, it's a tradeoff.

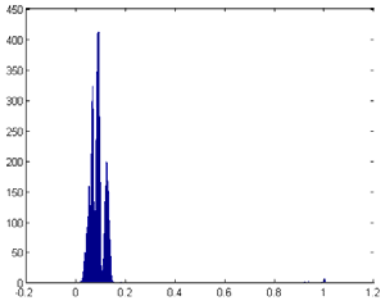
We just show how the two parameters affect the results. We can obtain much better results by other methods, so we won't give the exact number here.

5.2 Results of Kernel estimator

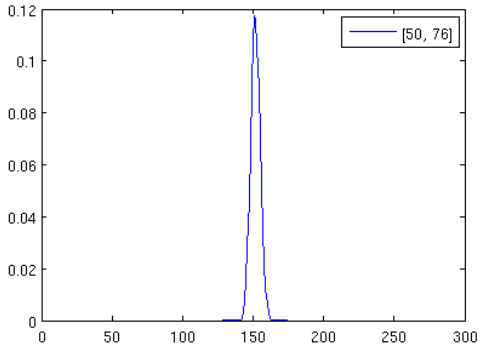
1. Obtained pdf. Of difference pixels

1) The pixel which remains beach, i.e., almost the static background, over some frames:

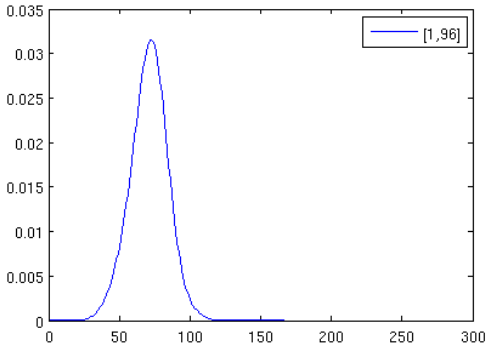
(a) Beach area without crabs crossing from frame #301~500; $x \in [1,160],\ y \in [71,170]$



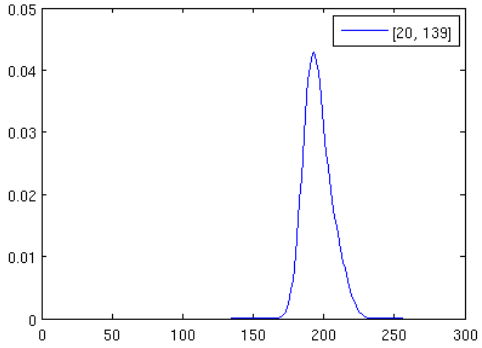
(b) Histogram of peak probability



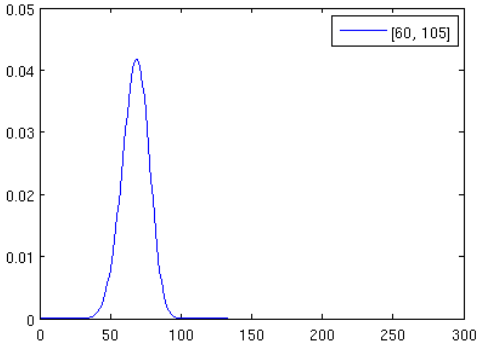
(c) pdf. of pixel [50,76] (always sand)



(d) pdf. of pixel [1,96]



(e) pdf. of pixel [20,139]



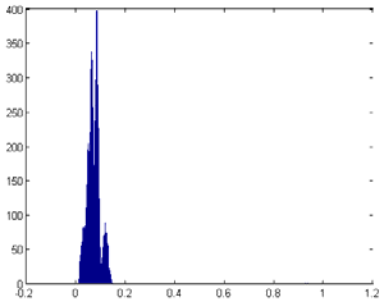
(f) pdf. of pixel [60,105]

Fig. 6. Estimator for static background without crabs moving across

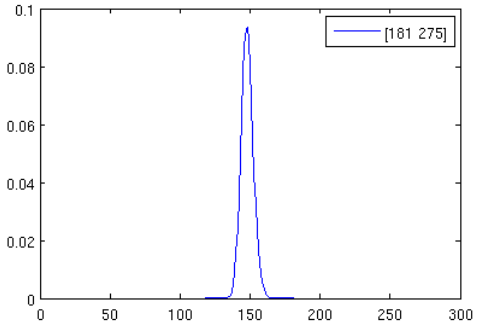
Locations of peak in each diagram are not the same due to the fact that beach is variegated. A strictly static background pixel could generate estimator that looks like (c) with peak up to 0.15.

2) The pixel which is beach for most of time, but some time it's occupied by a crab:

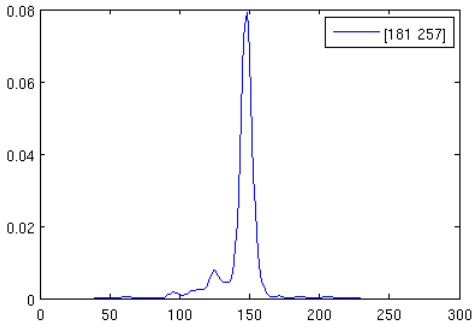
(a) A crab passing the beach.



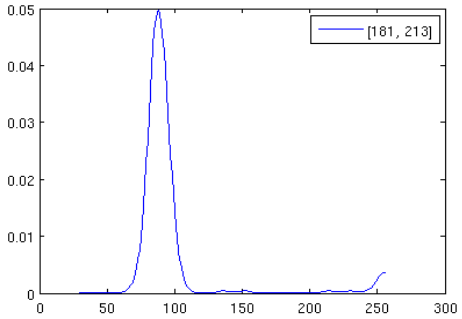
(b) Histogram of peak probability



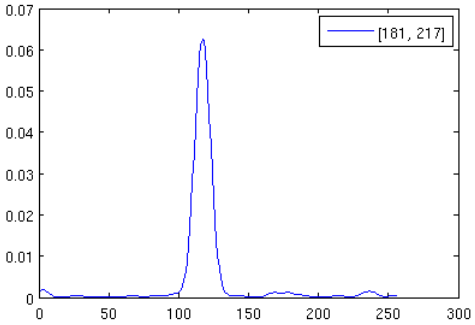
(c) pdf. of pixel [181,275] (always sand)



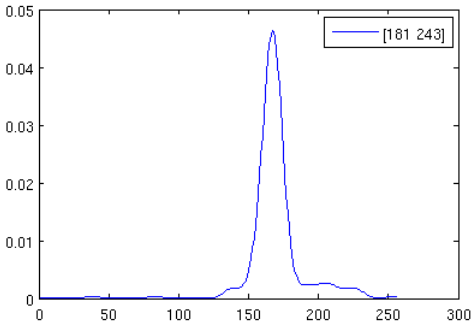
(d) pdf. of pixel [181,257] (sometimes crab)



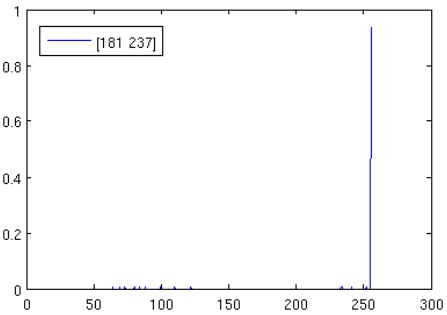
(e) pdf. of pixel [181,213]



(f) pdf. of pixel [181,217]



(g) pdf. of pixel [181,243]



(h) pdf. of pixel [181,237] (always crab's claw)

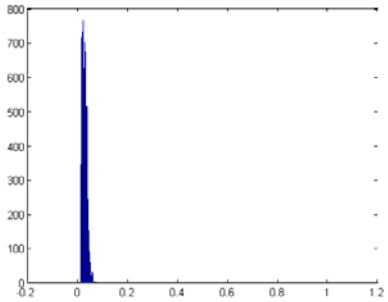
Fig. 7. Estimator for background with moving crab.

Estimator for pixels in the trace of crab are demonstrated in (d)~(g). As a comparison, we also show in (c) the estimator of a pixel which is not in the trace of crab. We can see that it's similar to those figures in Fig. 6.

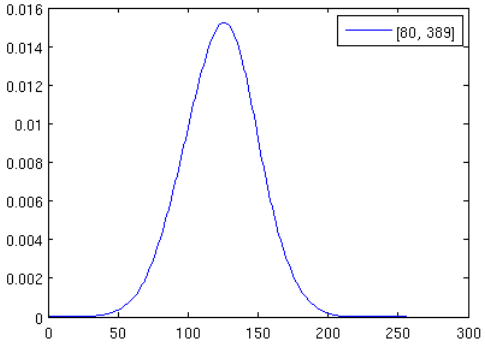
(h) is a special case: a fixed white pixel which is in the crab's claw.

3). The pixel which remains water area, i.e. dynamic background, over some frames

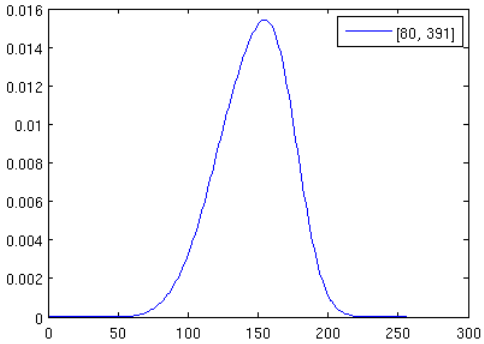
(a) Water area without crabs crossing from frame #301~500; $x \in [1,160],\ \ y \in [351,450]$



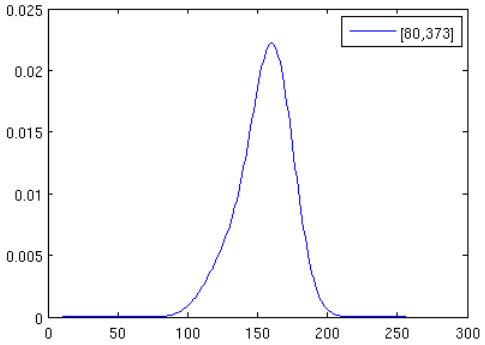
(b) Histogram of peak probability



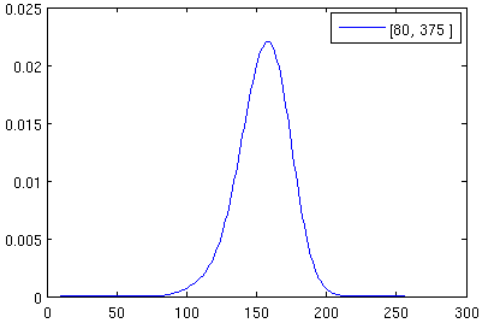
(c) pdf. of pixel [80,389]



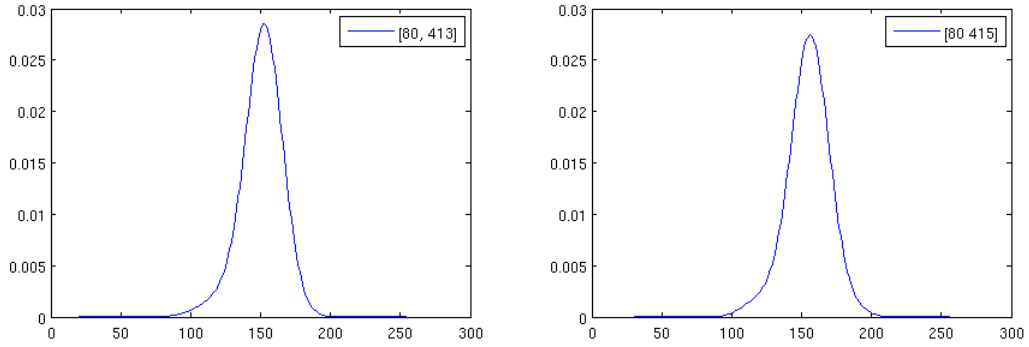
(d) pdf. of pixel [80,391]



(e) pdf. of pixel [80,373]



(f) pdf. of pixel [80,375]



(g) pdf. of pixel [80,413] (h) pdf. of pixel [80,415]

Fig. 8. Estimator for dynamic background (e.g. water).

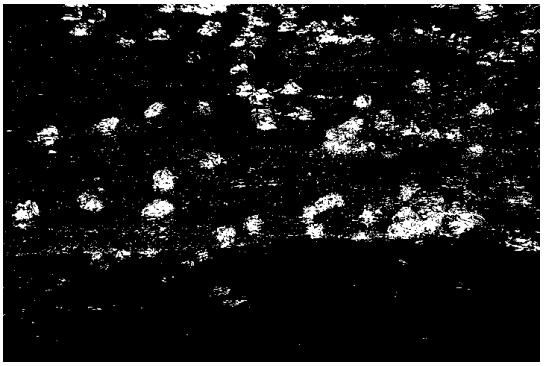
The pdf. are flatter than those of fixed background, and have much smaller peak probability, as shown in (b). The reason is that the luminance of water area varies more than the luminance of beach. The pixels with the same j coordinate are likely to share similar distribution and peak probability, as shown in (c)&(d), (e)&(f) and (g)&(h), because water ripple is mainly horizontal.

Based on those pdf. of typical pixels, we can decide our threshold T_m to be: below the probability density of typical sand and typical water, but above that of typical crabs.

2. Foreground detection

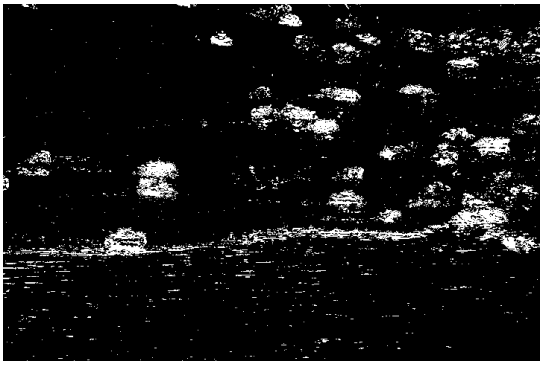
Comparison between the two detected foreground images, one is for the 10th frame with a lot of crabs, even overlapped, the other is the 500th frame with less crabs, each crab separated.

(a) # 10 frame from video crab1.avi



(b) Foreground of (a), $T_m = 0.002$

(c) # 500 frame from video crab1.avi



(d) Foreground of (c), $T_m = 0.002$

Fig. 9. Detected foreground, with background based on #301~400 frames

The water ripple in (a) doesn't appear in (b), which means we have a good estimation of the background, in spite that the computation is non-causal – observed image is the 10th frame, but background is computed based on the 301th ~400th frames. For the 500th frame, the noise caused by water ripple is also removed, but not as significantly as for the 10th frame. Thus, out counter can follow the change of the number of the crabs.

The "white line" appeared on (d) is caused by flux. From the video, we can observe that the water area suddenly extended at about the 500th frame, i.e., some "beach" pixels during the previous frames become "water area". The model

based on the 301th ~400th frames didn't expect this change in the background. As a result, there is a "while line" in the detected foreground image.

3. Suppression of false detection

Applying median filter and then morphological operation, imclose, to the detected foreground of the 780th frame (with 41 crabs, counted manually), we have:

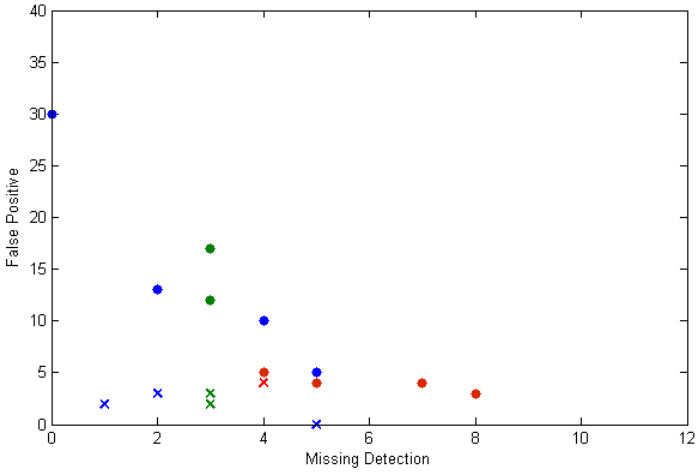


Fig. 10. The false detection and miss detection of the image, processed by median filter and closing operation

Medfilt	[11 11]	[11 9]	[11 7]	[11 5]	[9 7]	[9 5]	[9 3]	[9 1]	[7 7]	[7 5]
False Positive	3	4	4	5	5	10	13	30	12	17
Missing Detection	8	7	5	4	5	4	2	0	3	3
Number of Crabs	36	38	40	42	41	47	52	71	50	55
Difference with actual number (41)	-5	-3	-1	+1	0	+6	+11	+30	+9	+14

Table 1. #780 frame, result comparison for difference median filter masks

	False Positive		Missing Detection		Enumeration		Difference (actual number is 41)	
	Median	After Closing	Median	After Closing	Median	After Closing	Median	After Closing
[11 5]	5	4	4	4	42	41	+1	0
[9 1]	30	2	0	1	70	40	+30	-1
[9 3]	13	3	2	2	52	42	+11	+1
[9 7]	5	0	5	5	41	36	+0	-5
[7 5]	17	3	3	3	55	41	+14	0
[7 7]	12	2	3	3	50	40	+9	-1

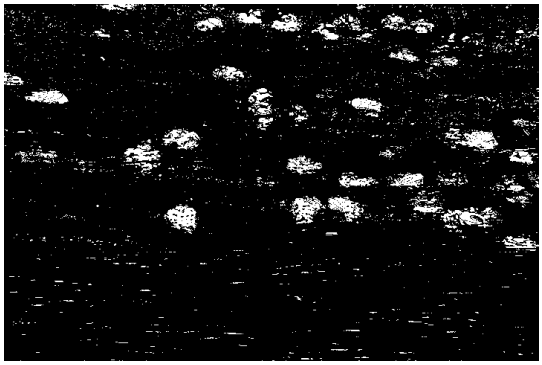
Table 2. Imclose, using structuring element disk(6 4)

From Fig. 10, we can see how the parameters of median filter and morphological operation may affect the results.

As we adjust the parameters, the false detection and miss detection change dependently. It's a trade-off. The optimal parameters are those lead to minimum false detection and minimum miss detection at the same time, i.e., false detection 3, miss detection 2, with median filter mask [9,3] and closing operation using structuring element disk(6,6). Since in Table 1, Table 2 and Fig. 10. we have shown the result of several experiments with different parameters, here we omit most of the corresponding figures. Instead we give the figures of the best result in Fig. 11.

We have also tried the third method to suppress the false detection, but it doesn't lead to better result than median filter and morphological operation. So we omit those results too.

(a) Original: #780 frame



(b) Foreground of (a), $T_m = 0.0008$

(c) Median filtered (b), filter mask [9,3]

(d) Closing with element disk[6 6] of (c)

(e) Red: false detection; blue: missing detection

(f) Manually counted crabs

Fig. 11. Post-processed foreground images

6 Conclusions

We have implemented a method (Kernel density estimator, with Normal distribution as Kernel function) to detect the presence of the crabs and then count their number. Based on this method, and some post-processing such as median filtering, morphological operations, we can remove most of the noise caused by water ripple. The number of crabs computed by the program is close to the real number. Since we have made some ideal assumptions for our work, such as no camera oscillation, which are probably difficult to be satisfied, error is inevitable, but it's small and acceptable.

7 Appendix

```
%%%%%%%%%Background subtraction using Median Background%%%%
% median background building based on previous 500 frames
% X500.mat contains the previous 500 Frames
load X500.mat
med500=median(X500, 3);

mov=aviread('crab1.avi',501);
I=rgb2gray(mov.cdata);
II=abs(uint8(I)-med500);

bw=im2bw(II, 0.1); %graylevel thresholding 0.1
L=bwlabel(bw, 4); %4-neighbor labeling
max_num=max(max(L)); %number of detected objects
goal=zeros(480,720);
t=0;
for i=1:max_num
      Z=find(L==i);
      if length(Z)>20 %thresholding the minumum number of connected elements
in an object
            t=t+1;
            goal(Z)=t;
      end
end
%t contains the final detected number of crabs

%%%%%%%%%%%%%%%%Building look-up table%%%%%%%%%%%%
S=zeros(256, 256, 256);

for k=1:256 % changing variance
      S(k, k, 1)=1;
end

for s=2:256
      for i=1:256 % chaning mean
            S(i, :, s)=pdf('Normal', 1:256, i-1, s-1);
      end
end

%%%%%%%%%%Building Mixture of Gaussian Model%%%%%%%%%%
% X.mat contains previous 100 frames used in building the model
% NewG.mat is the built model using mixture of Gaussian Method
Fnum=100;
NewG=zeros(480*720, 256);

for i=1:Fnum-1
```

```
    B(:, :, i)=abs(X(:, :,i)-X(:, :,i+1));
end

ss=round(median(B, 3)/(0.68*sqrt(2))); % ss for each pixel

for i=1:480
    for j=1:720
        for k=1:Fnum
            NewG(720*(i-1)+j, :)=NewG(720*(i-1)+j, :)+S(X(i, j, k)+1, :, ss(i,
j)+1);
        end
    end
end

NewG=NewG/Fnum;

%%%%%%%%%%Foreground Detection%%%%%%%%%%%%%%%%%%
load NewG.mat
thre=0.0008;
mov=aviread('tifb.avi',780);
I=double(rgb2gray(mov.cdata));

for i=1:480
    for j=1:720
        if NewG(720*(i-1)+j, I(i, j)+1)>thre
            T(i,j)=0; % background
        else
            T(i,j)=255; % foreground
        end
    end
end

A=medfilt2(T, [9 1]);
figure, imshow(A, []);
bw=im2bw(A);
L=bwlabel(bw, 4);
maxnum=max(max(L));

goal=zeros(480, 720);
t=0;

for i=1:maxnum
    Z=find(L==i);
    if length(Z)>10
        t=t+1;
        goal(Z)=t;
```

```
    end
end

figure(1), imshow(goal);

SE1=strel('disk', 6, 4);
im1=imclose(A, SE1);
L1=bwlabel(im1, 4);
maxnum1=max(max(L1));
goal1=zeros(480, 720);
t1=0;
for i=1:maxnum1
    Z=find(L1==i);
    if length(Z)>10
        t1=t1+1;
        goal1(Z)=t1;
    end
end

figure(2), imshow(goal1);
```

References

[1] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, “Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance”, IEEE, VOL. 90, No. 7, pp. 1152-1163, July 2002.

[2] N. Friedman, and S. Russell, “Image Segmentation in Video Sequences: A Probabilistic Approach”, in: The 13th Conference on Uncertainty in Artificial Intelligence(UAI 97), 1997.

[3] A. Elgammal, D. Harwood, L. Davis, “Non-parametric Model for Background Subtraction”, in: The 6th European Conference on Computer Vision. Dublin, Ireland, June/July 2000.

[4] Y. Sheikh, and M. Shah, “Bayesian Modeling of Dynamic Scenes for Object Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, VOL. 27, No. 11, pp. 1778-1792, 2005.

[5] Massimo Piccardi, “Background Subtraction Techniques: A Review”, in: The ARC Centre of Excellence for Autonomous System, UTS, April, 2004.

[6] J. Migdal, and W. Eric L. Grimson, “Background Sbtraction Using Markov Thresholds”, in: Proc. IEEE Workshop on Motion and Video Computing, 2005.