CO227: Computer Engineering Project : Final Report

FOREGROUND ESTIMATION IN DYNAMIC BACKGROUND CONDITIONS

Project Team

E/14/158 Gihan Jayatilaka
E/14/339 Suren Sritharan
E/14/379 Harshana Weligampola

Instructor in charge:

Mr. Brian Udugamuwa

Supervisors

Dr. Dhammila Elkaduwe
Dr. Roshan Godaliyadda
Dr. Vijitha Herath
Dr. Parakrama Ekanayeka
Dr. Nalin Harishchandra

CONTENTS

1. Introduction and background

Motion detection plays a fundamental function in any object tracking or video surveillance algorithm. Almost all such algorithms begin with motion detection. The initial background removal and foreground detection plays a crucial role in surveillance since the output of all subsequent processing would depend on performance and reliability of the initial algorithm. However, detecting regions of change in images of the same scene is not a straightforward task since it does not only depend on the features of the foreground elements, but also on the characteristics of the background such as, for instance, in the presence of dynamic background elements such as trees.
Therefore our goal was to initially solve the base case scenario with static background and then move onto cases with dynamic background and finally
Our main goal was to detect motion in aquatic based environments which could prove vital for applications such as lifeguards, coastguards, etc.
So, after analyzing previous related approaches, we propose a motion algorithm that successfully deals with all the arisen problems.
Finally, performance evaluation, analysis, and discussion are carried out.

2. Problem definition and proposed solution

2.1. Problem definition:
“Foreground estimation and background subtraction for dynamic background conditions.”

2.2. Problem explanation

Every video feed or photograph has two sections as background and foreground. The foreground is the set of areas we are interested in (ex: People, Vehicles) and the background is the complement of it. Even though it is straightforward to detect the foreground in static backgrounds because all that moves is the foreground, it is challenging to detect the foreground in dynamic background conditions such as outdoors where trees shake to wind and aquatic scenes where water ripples.

2.3. Proposed solution

Since there is no clear way of detecting the foreground perfectly on any dynamic background condition, we try to estimate the foreground. The approach is to estimate the probability of every pixel being foreground at a given time and use that probability density distribution as a mask for the video to estimate the foregrounds. There are numerous algorithms already proposed for this problem but they work only on specific conditions.

Some work only on constant lighting conditions

Some work only on shaking trees, some work only on aquatic scenes.

We try to analyze these algorithms and compare results in this project. This include adding our own improvements to the existing algorithms to improve their accuracy, performance and also to generalize them to work on different conditions. We will be proposing the most general algorithm for the problem. Finally, the solution will be wrapped as a shell application for easy usage.

The algorithms we use are as follows,

1. Pixel based adaptive segmentation
2. Mixture models
    a. Gaussian mixture model
    b. Cylindrical model
3. Adaptive mixture models
    a. Adaptive gaussian mixture model
    b. Adaptive cylindrical mixture model
4. Robust principal component analysis
5. Hierarchical Video Segmentation

2.3.1. Pixel based adaptive segmentation

In PBAS several parameters are adaptively adjusted at runtime for each pixel separately. Background model contains N recently observed pixel values, B(xi) = {B1(xi), . . . , Bk(xi), . . . , BN(xi)}. Objective is to update these values based on the the decision from the input frame values I(xi) and parameters for each pixel.

Parameters for given pixel are, per-pixel threshold R(xi) and per-pixel learning parameter T(xi). A pixel xi is decided to belong to the background, if its pixel value I(xi) is closer than a certain decision threshold R(xi) to at least #min of the N background values. Thus, the foreground segmentation mask is calculated as,

$$F(x_i) = \begin{cases} 1 & \#\{dist(I(x_i), B_k(x_i)) < R(x_i)\} < \#_{min} \\ 0 & \text{else} \end{cases}$$

Here, F = 1 implies foreground. It can thus be seen, that the decision making involves two parameters: (1) The distance threshold R(xi), which is defined for each pixel separately and which can change dynamically; and (2) the minimum number #min, which is a fixed global parameter.

If F(xi) = 0, the current pixel is adapted to background model with a probability p = 1/T(xi). A pixel is adapted by randomly replacing a given pixel in B(xi) set.

R(xi) parameter is updated as,

$$R(x_i) = \begin{cases} R(x_i) \cdot (1 - R_{inc/dec}), \text{ if } & R(x_i) > \bar{d}_{min}(x_i) \cdot R_{scale} \\ R(x_i) \cdot (1 + R_{inc/dec}), \text{ else} & \end{cases}$$

Where Rinc/dec and Rscale are constants. dmin(xi) is the average of minimum distance for I(xi) and Bk(xi) for the past history.
T(xi) is updated as,

$$T(x_i) = \begin{cases} T(x_i) + \frac{T_{inc}}{\bar{d}_{min}(x_i)}, & \text{if } F(x_i) = 1 \\ T(x_i) - \frac{T_{dec}}{\bar{d}_{min}(x_i)}, & \text{if } F(x_i) = 0 \end{cases}$$

Tinc and Tdec are constants.

2.3.2. Mixture models

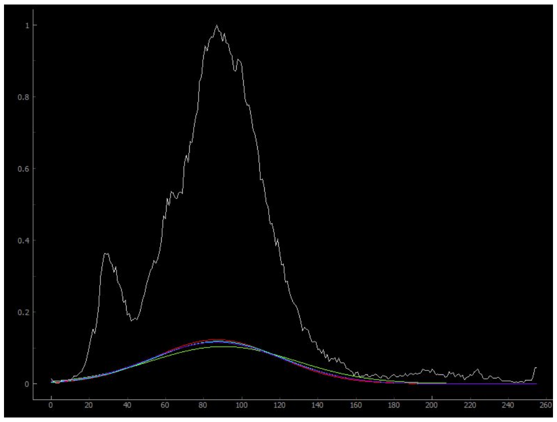
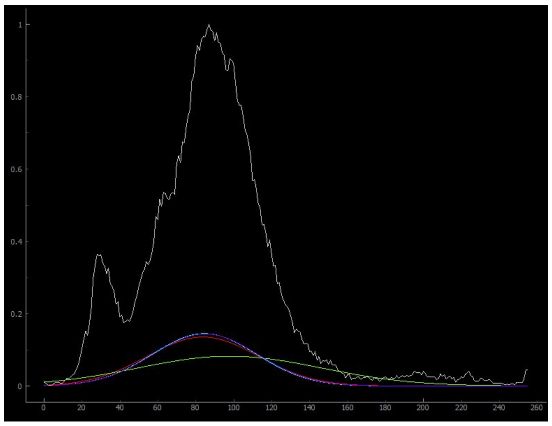
In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs.
In foreground estimation problems, every pixel at (x,y) - Pi(x,y) at a given time t, P(x,y,t) is considered to be a mixture of clusters {C1,C2,C3…} by probabilities Pr(Pi(x,y,t)=C1,Pr(Pi(x,y,t)=C2,.....

These probabilities are estimated using Expectation Maximization (EM) algorithm and then the clusters are split as to background or foreground on their parameters such as variance, normality, entropy.
Finally, Pr(Pi(x,y,t)) belonging to foreground or background is calculated using all these parameters.

2.3.2.1. Expectation Maximization Algorithm

In statistics, an expectation–maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step.

2.3.2.2. Gaussian Mixture Model
Here the clusters are assumed to be gaussian. The EM algorithm is initialized with values for the cluster parameters (since this is gaussian, the means and variances). Iteratively the parameters are fit.

	
Initialization	After 5 iterations
Figure 01: How the cluster parameters adjust with iterations of EM algorithm	

2.3.2.2. Cylindrical Mixture Model
Here the probability distributions of clusters is assumed to be cylindrical. The algorithm is similar to GMM.

2.3.3. Adaptive models

The mixture models have a few shortcomings as they cannot handle
Lighting changes
Long term changes of the surrounding
And also the mixture models running on EM algorithm needs to keep track of all the values of the time series of a pixel throughout every iteration.
The adaptive mixture models are the solution to all these problems. They “adapt” the cluster parameters according to every frame.

2.3.3.1 AGMM

An on-line K-means approximation is used to update the Gaussians.
If a new pixel value, $\mathbf{X_{t+1}}$, can be matched to one of the existing Gaussians (within $\mathbf{2.5\sigma}$), that Gaussian's $\boldsymbol{\mu_{i,t+1}}$ and $\boldsymbol{\sigma^2_{i,t+1}}$ are updated as follows:

$$\mu_{i,t+1} = (1 - \rho)\mu_{i,t} + \rho X_{t+1}$$

$$\sigma^2_{i,t+1} = (1 - \rho)\sigma^2_{i,t} + \rho(X_{t+1} - \mu_{i,t+1})^2$$

where $\boldsymbol{\rho = \alpha N (X_{t+1}|\mu_{i,t} , \sigma^2_{i,t})}$ and $\boldsymbol{\alpha}$ is a learning rate.

AGMM could be further modified adding the following steps. (This was not done in this project.)

Prior weights of all Gaussians are adjusted as follows:
$\boldsymbol{\omega_{i,t+1} = (1 - \alpha)\omega_{i,t} + \alpha(M_{i,t+1})}$
where $\mathbf{M_{i,t+1} = 1}$ for the matching Gaussian and $\mathbf{M_{i,t+1} = 0}$ for all the others.
If $\mathbf{X_{t+1}}$ do not match to any of the K existing Gaussians, the least probable distribution is replaced with a new one. Least probably" in the ω/σ sense.
New distribution has $\boldsymbol{\mu_{t+1} = X_{t+1}}$, a high variance and a low prior weight.

2.3.3.2 ACMM

ACMM works same as AGMM but with a cylindrical probability distribution for clusters instead of the spherical ones. The cluster parameters are the radius, length and the orientation of the cylinders.
The cylinder's' radius and length are adapted to the data points in the time series of the pixel just as AGMM is adapting the spheres. The orientation of the cylinder is adapted by using PCA (principal component analysis) technique.

The adaptation takes place as per

$$L^2_{k,t} = (1-\rho)L^2{}_{k,t-1} + \rho {L_1}^2,$$

$$R^2_{k,t} = (1-\rho)R^2{}_{k,t-1} + \rho {R_1}^2$$

Similar to AGMM and here L is the half length of a cylinder, R is the radius of a cylinder.

Here the **M**, (1 if a data point belongs to a cluster) is taken with the conditions

$${L_1}^2 < D_1.L^2{}_{k,t-1} \qquad {R_1}^2 < D_2.R^2{}_{k,t-1}$$

2.3.4 RPCA

RPCA decompose matrix D into a low-rank term and a sparse term by solving the following convex problem:

$$\min_{A,E} \|A\|_* + \lambda\|E\|_1 , \ s.t. \ D = A + E,$$

where $\|A\|_*$ is the nuclear norm of a matrix, $\|E\|_1$ denotes the `1 norm of a matrix seen as a long vector, and λ > 0 is a parameter. The low-rank and sparse components correspond to the static background and the moving objects, respectively
The lower rank emphasizes that along the time axis two background pixels are linearly dependant since the variation in factors such as light intensity would affect all background pixels similarly.

2.3.5 HVS

Hierarchical video segmentation uses the concept of MST to construct segments. MST's are clustered based on both hard limits and soft limits. By performing the clustering hierarchically it can be made sure that both the spatial localities as well as the colour differences can be given equal weights.

3. Design and Implementation of the solution

3.1 Design

The solution design was done with the help of

Literature from IEEE xplore
Project meetings with the product owner
Scrum meetings
Supervisor meetings

3.2. Implementation

The following were used in the implementation of algorithms

Programming languages	Python, MATLAB
Numerical computing packages	Numpy
Scientific computing packages	Scipy
Plotters	Matplotlib, PyQtGraph
Code sharing	github
Video input output	opencv
Computation	Aiken.ce.pdn.ac.lk
Shell application	python

4. Results

The project analyzed the performance of all the approaches on a data set with a variety of background and foreground conditions.

The data set used has

- Static background
- Dynamic background
  - Shaking trees
  - Rippling water
- Foreground objects
  - People
  - Vehicles
  - Boats

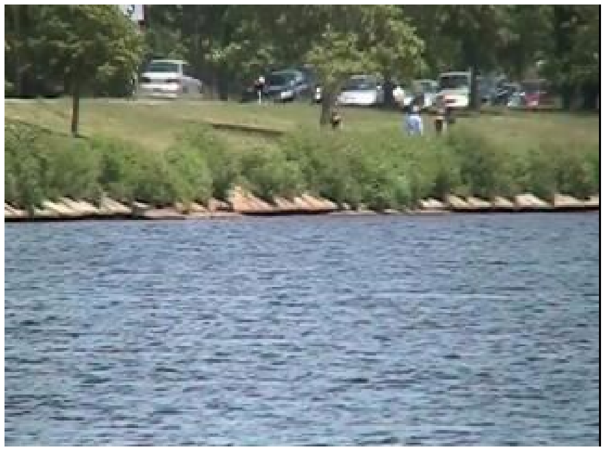
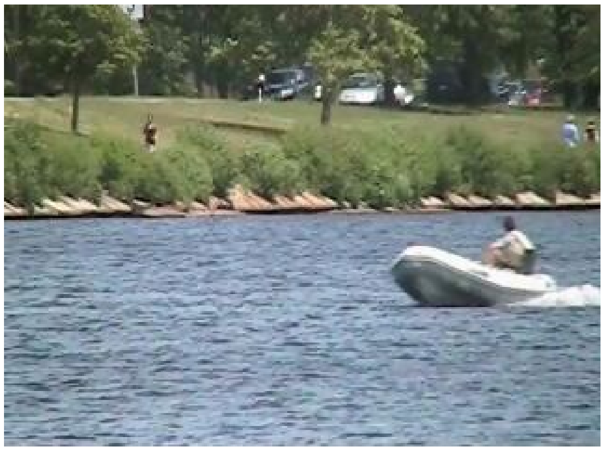
	

Figure 02.a: Test video frame with people and vehicles as foreground objects with trees and water as dynamic background	Figure 02.b: Test video frame with people vehicles and a boat as foreground objects with trees and water as dynamic background

The actual foreground and background is below. This is human decision on which part is foreground. A perfect algorithm would have got the same result.

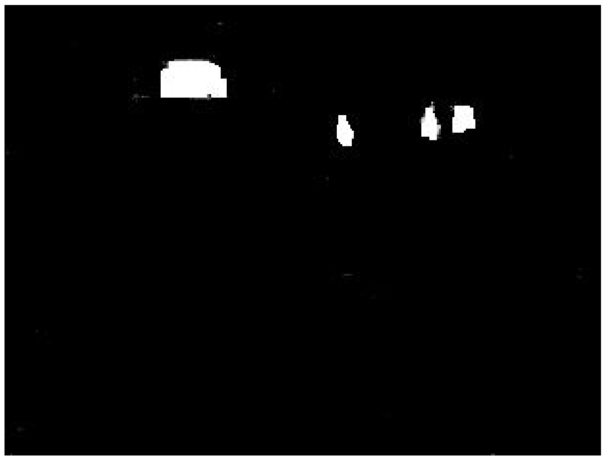
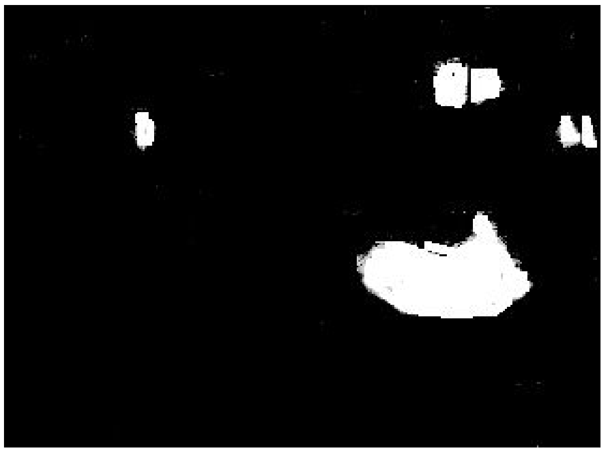
	

Figure 03.a: The actual background (black) and foreground (white) of the figure (a)	Figure 03.b: The actual background (black) and foreground (white) of the figure (b)

4.1 Results from PBAS

4.1.1. Results from PBAS with a static limit for choosing background

Did not give consistent results for the videos. There was a higher amount of noise in most frames.

4.1.2. Results from PBAS choosing background with respect to variance of the behavior

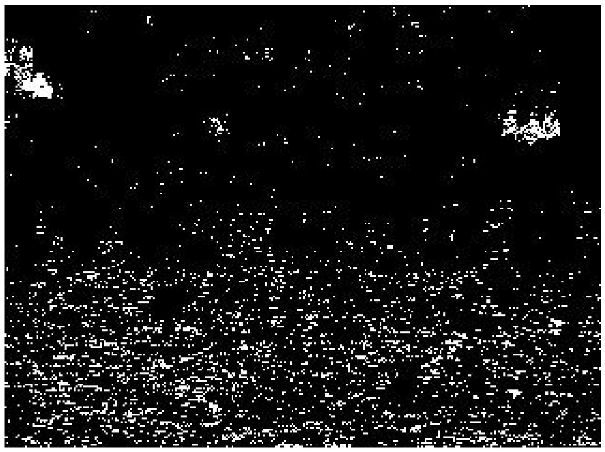
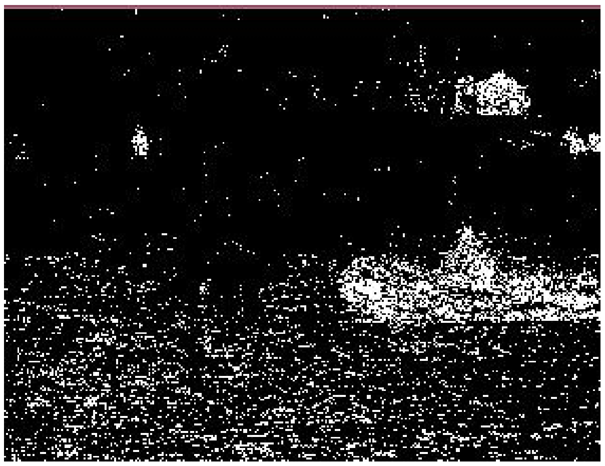
	

Figure 04.a: The background (black) and foreground (white) of the figure (a) according to PBAS algorithm choosing background with respect to variance of the behavior	Figure 04.b: The background (black) and foreground (white) of the figure (b) according to PBAS algorithm choosing background with respect to variance of the behavior

4.2 Results from GMM

GMM algorithm with hard limits for variance of clusters in choosing background and foreground.

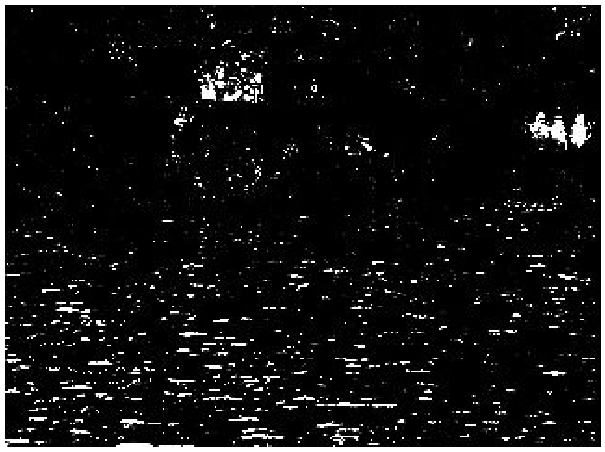
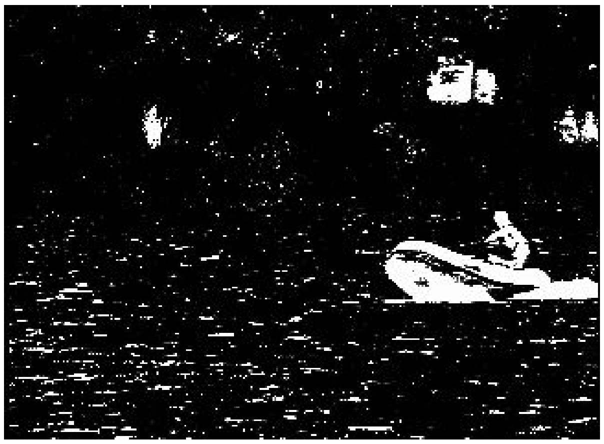
	

Figure 05.a: The background (black) and foreground (white) of the figure (a) according to GMM algorithm	Figure 05.b: The background (black) and foreground (white) of the figure (b) according to GMM algorithm

The other approaches for splitting clusters between background and foreground

Normality measure

Entropy measure

did not give any good results.

4.3. Results from AGMM

4.3.1. Results from text book AGMM

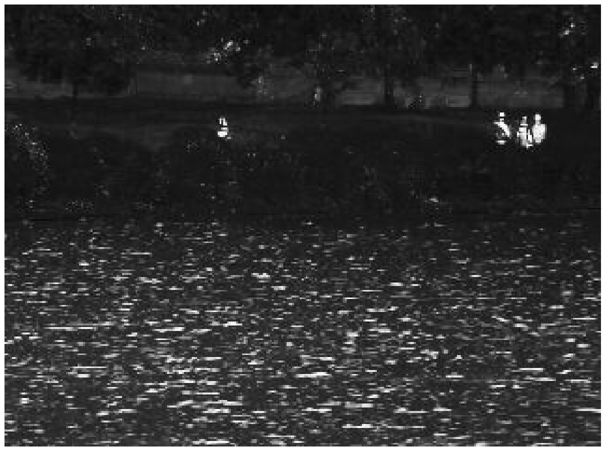
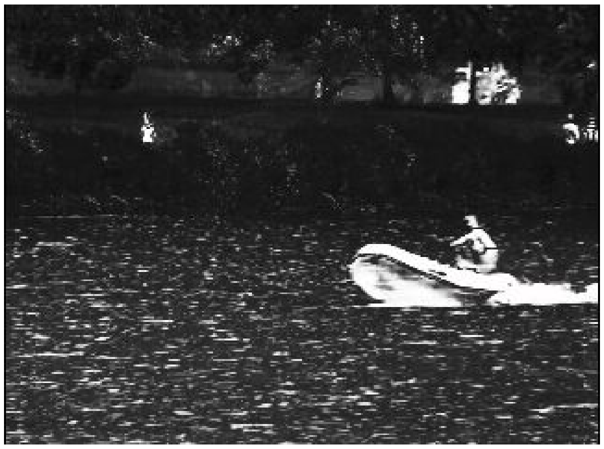
	

Figure 06.a: The background (black) and foreground (white) of the figure (a) according to AGMM algorithm	Figure 06.b: The background (black) and foreground (white) of the figure (b) according to AGMM algorithm

4.3.1. Results from two layer AGMM

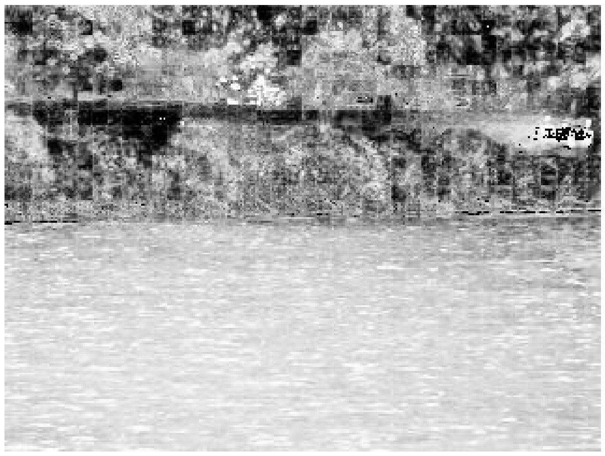
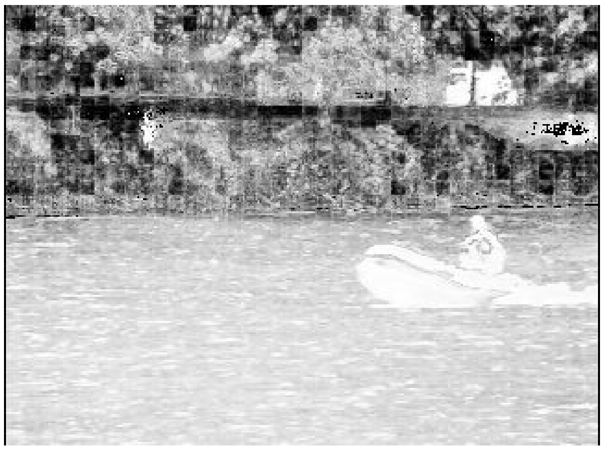
	

Figure 07.a: The background (black) and foreground (white) of the figure (a) according to two layer AGMM algorithm	Figure 07.b: The background (black) and foreground (white) of the figure (b) according to two layer AGMM algorithm

4.3. Results from ACMM

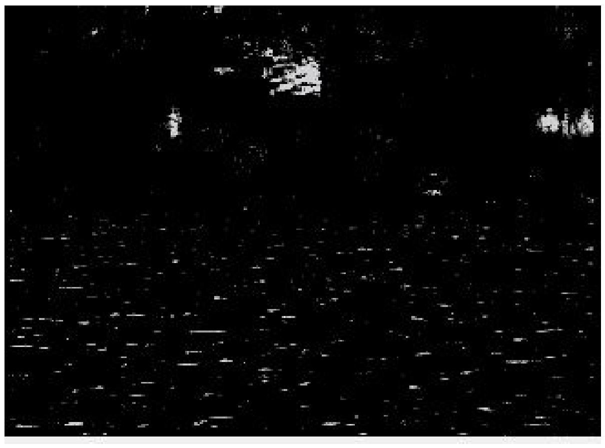
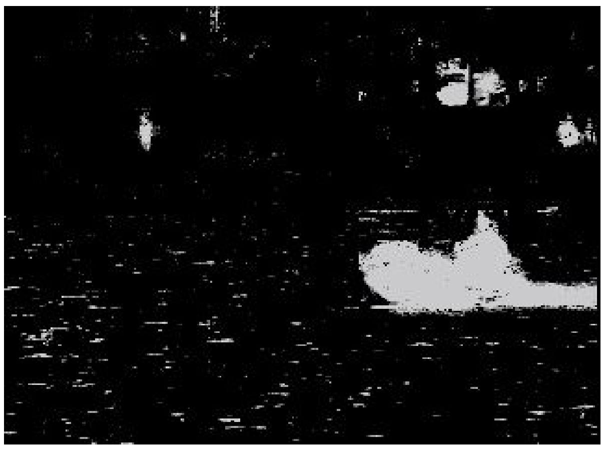
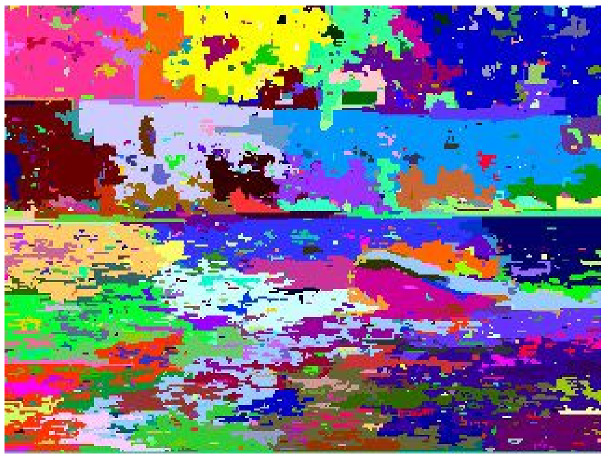
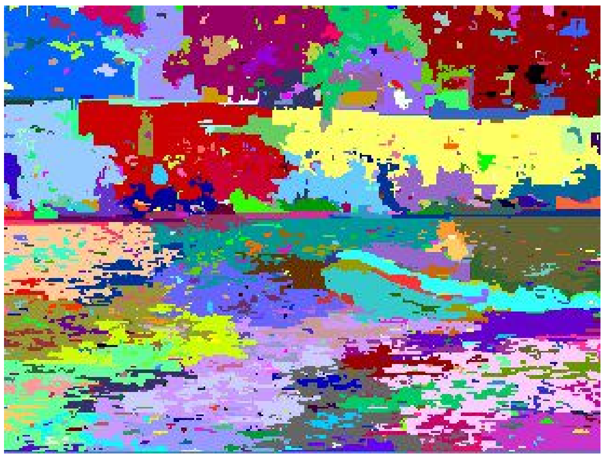
	

Figure 08.a: The background (black) and foreground (white) of the figure (a) according to ACMM algorithm	Figure 08.b: The background (black) and foreground (white) of the figure (b) according to ACMM algorithm

4.4 Results from HVS

HVS algorithm could be fine tuned to split the video into different segments. Some segments had foreground objects and some had background. There was minimum overlap of two kinds in any segment.

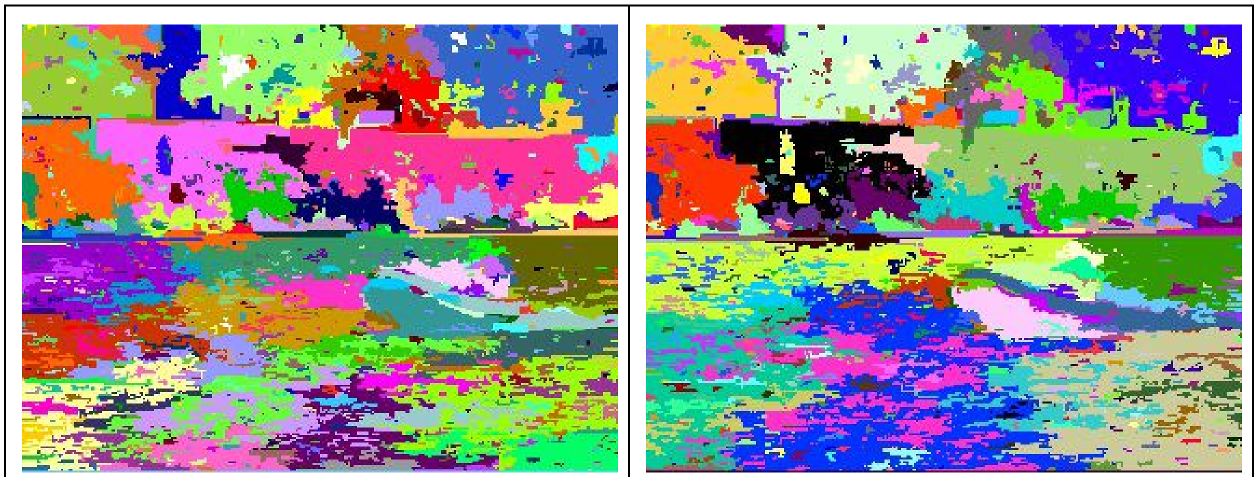


Figure 09. Hierarchical video segmentation

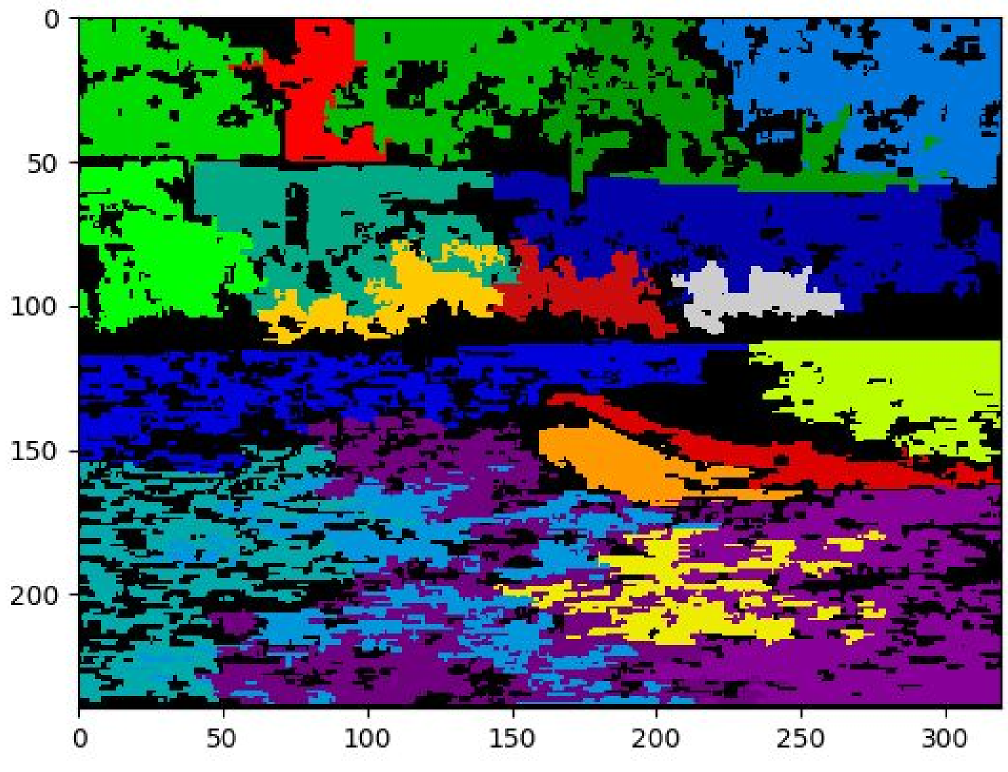


Figure 10: Segments obtained by HVS

4.4 Results from RPCA

Robust principal component analysis decomposes the video into sparse and low rank matrix. But due to the inability to run it in real time and due to performance constraints with longer videos, further analysis was not done.

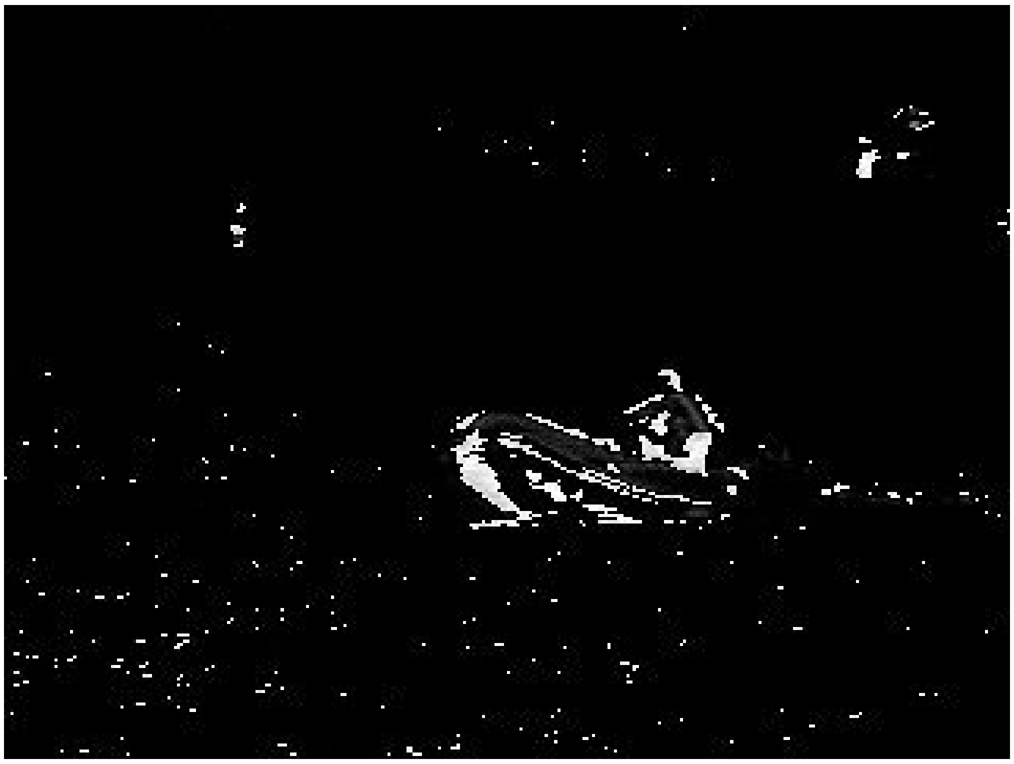
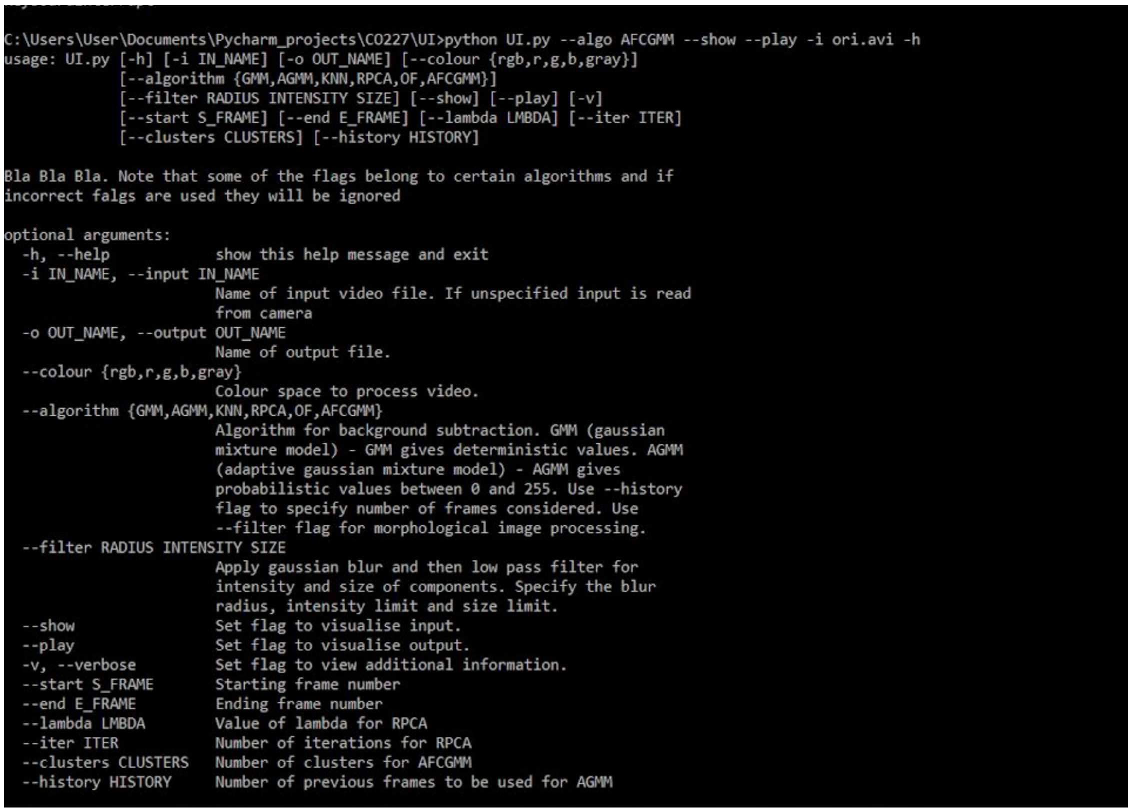


Figure 11: Results from RPCA

4.5 Shell application

A shell application was developed to make it easier for users to try these algorithms on videos. It is capable of taking input videos or taking the webcam as a video input and to output the results from any algorithm. It supports multiple functionalities including tunable parameters, ability to read and write videos and includes algorithms such as GMM, AGMM, AFCMM, RPCA, etc.
A sequence of algorithms specifically hand picked for this video gave the best result. The input was initially processed using AGMM followed by the cylindrical model algorithm and finally applying morphological image processing using gaussian blur and size filters. These parameters can be fine tuned depending on different scenarios.



```
C:\Users\User\Documents\Pycharm_projects\CO227\UI>python UI.py --algo AFCGMM --show --play -i ori.avi -h
usage: UI.py [-h] [-i IN_NAME] [-o OUT_NAME] [--colour {rgb,r,g,b,gray}]
             [--algorithm {GMM,AGMM,KNN,RPCA,OF,AFCGMM}]
             [--filter RADIUS INTENSITY SIZE] [--show] [--play] [-v]
             [--start S_FRAME] [--end E_FRAME] [--lambda LMBDA] [--iter ITER]
             [--clusters CLUSTERS] [--history HISTORY]

Bla Bla Bla. Note that some of the flags belong to certain algorithms and if
incorrect falgs are used they will be ignored

optional arguments:
  -h, --help            show this help message and exit
  -i IN_NAME, --input IN_NAME
                        Name of input video file. If unspecified input is read
                        from camera
  -o OUT_NAME, --output OUT_NAME
                        Name of output file.
  --colour {rgb,r,g,b,gray}
                        Colour space to process video.
  --algorithm {GMM,AGMM,KNN,RPCA,OF,AFCGMM}
                        Algorithm for background subtraction. GMM (gaussian
                        mixture model) - GMM gives deterministic values. AGMM
                        (adaptive gaussian mixture model) - AGMM gives
                        probabilistic values between 0 and 255. Use --history
                        flag to specify number of frames considered. Use
                        --filter flag for morphological image processing.
  --filter RADIUS INTENSITY SIZE
                        Apply gaussian blur and then low pass filter for
                        intensity and size of components. Specify the blur
                        radius, intensity limit and size limit.
  --show                Set flag to visualise input.
  --play                Set flag to visualise output.
  -v, --verbose         Set flag to view additional information.
  --start S_FRAME       Starting frame number
  --end E_FRAME         Ending frame number
  --lambda LMBDA        Value of lambda for RPCA
  --iter ITER           Number of iterations for RPCA
  --clusters CLUSTERS   Number of clusters for AFCGMM
  --history HISTORY     Number of previous frames to be used for AGMM
```

Figure 12: Screenshots from the shell application

5. Conclusions and future work

5.1 Conclusions

The results from all approaches show that,

- PBAS is suitable only for static backgrounds or dynamic backgrounds with minimum changes.
- GMM’s cluster classification into foregrounds and backgrounds should be done considering variances. Normality and entropy are not good indicators.
- Adaptive mixture models performs better than EM algorithm in almost every case.
- Two layer GMM is not a good approach.
- Cylindrical models are more accurate than spherical gaussian models when it comes to identifying water as a dynamic background.
- HVS is a good segmentation algorithm for videos but it is difficult to split the segments into background and foreground without other metrics.
- RPCA is accurate but requires huge amounts of RAM and processing time so therefore not a suitable algorithm without high end servers.
- A sequence of algorithms can give a good result for a particular video, but this sequence and the filters to be used can vary from case to case. It is a matter of fine tuning if you want to get a very accurate result for a particular video.
- The shell application makes it easy for users to experiment different algorithms, algorithm sequences and filters on a video.

5.2 Future work

- Parallel implementations of the algorithms.
- To apply a set of algorithms as a sensor fusion instead of a sequence.
- Neural gas and Self Organizing Maps to cluster the time series of a pixel.
- Graph algorithms to do the filtering (Connected component algorithms)
- ObjectFlow to integrate movements in order to separate merging objects.

6. References

1. Adaptive Free Cylindrical Mixture Model for Foreground Estimation in Rapidly Fluctuating Dynamic Background Conditions. M.G.S. Jayasinghe, W.S.K. Fernando, A.A. Senerath, M.P.B. Ekanayake and G.M.R. Godaliyadda.
2. Efficient Hierarchical Graph-Based Video Segmentation. Matthias Grundmann, Vivek Kwatra, Mei Han, Irfan Essa.
3. Background Subtraction - Birgi Tamersoy
4. Adaptive background mixture models for real-time tracking. Chris Stauffer, W.E.L Grimson.
5. Detection of Moving Object in Dynamic Background Using Gaussian Max-Pooling and Segmentation Constrained RPCA. Yang Li, Guangcan Liu.
6. The Pixel-Based Adaptive Segmenter. M Hofmann.