

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE



## **Maritime Tracking Using Level Sets with Shape Priors**

**By:**

**Duncan Peter Frost**

**207510287**

**Supervisor:**

Professor Jules-Raymond Tapamo

**Co-Supervisors:**

Professor Roger Peplow

Mr Asheer Kasar Bachoo

In fulfilment of the academic requirements for the degree of Master of Science in Engineering, School of  
Engineering, University of KwaZulu-Natal

1 August 2012

## Abstract

Piracy is still a significant threat to ships in a maritime environment. Areas such as the coast of Somalia and the Strait of Malacca are still plagued by pirates, and the total international cost of piracy numbers in the billions of dollars. The first line of defence against these threats is early detection and thus maritime surveillance has become an increasingly important task over the years. While surveillance has traditionally been a manual task using crew members in lookout positions on parts of the ship, much work is being done to automate this task using digital cameras equipped with computer vision software. While these systems are beneficial in that they do not grow tired like their human counterparts, the maritime environment is a challenging task for computer vision systems. This dissertation aims to address some of these challenges by presenting a system that is able to use prior knowledge of an object's shape to aid in detection and tracking of the object. Additionally, it aims to test this system under various environmental conditions (such as weather). The system is based around the segmentation technique known as the *level set method*, which uses a contour in the image that is evolved to separate regions of interest. The system is split into two parts, comprising of an object detection stage that initially finds objects in a scene, and an object tracking stage that tracks detected objects for the rest of the sequence. The object detection stage uses a kernel density estimation-based background subtraction and a binary image level set filter, while the object tracker makes use of a tracking level set algorithm for its functionality. The object detector was tested using a group of 4 sequences, of which it was able to find a prior-known object in 3. The object tracker was tested on a group of 10 sequences for 300 frames a sequence. In 6 of these sequences the object tracker was able to successfully track the object in every single frame. It is shown that the developed video tracking system outperforms level set-based systems that don't use prior shape knowledge, working well even where these systems fail.

## **Preface**

The research discussed in this dissertation was done at the University of KwaZulu-Natal, Durban from February 2011 until July 2012 by Duncan Frost under the supervision of Professor Jules-Raymond Tapamo and co-supervision of Professor Roger Peplow and Mr Asheer Kasar Bachoo.

## **Declaration – Supervisor**

As the candidate's supervisor I agree to the submission of this dissertation.

---

Prof Jules-Raymond Tapamo

## **Declaration – Co-Supervisor**

As the candidate's co-supervisor I agree to the submission of this dissertation.

---

Prof Roger Peplow

## Declaration – Plagiarism

I, Duncan Frost, declare that

- i. The research reported in this dissertation/thesis, except where otherwise indicated, is my original work.
- ii. This dissertation/thesis has not been submitted for any degree or examination at any other university.
- iii. This dissertation/thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
- iv. This dissertation/thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
  - a) Their words have been re-written but the general information attributed to them has been referenced;
  - b) Where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
- v. Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.
- vi. This dissertation/thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation/thesis and in the References sections.



---

Duncan Frost

## Declaration – Publications

Details of contribution to publications that form part and/or include research in this dissertation:

Publication 1: D. Frost and J-R Tapamo. Maritime Tracking Using Level Sets with Shape Priors, **Sensors**.

(Journal paper submitted)



---

Duncan Frost

## **Acknowledgements**

I would like to thank my supervisor, Professor Jules-Raymond Tapamo and my co-supervisors Professor Roger Peplow and Mr Asheer Kasar Bachoo for their advice and guidance.

I also wish to thank my family for their continued support and patience throughout this process. No number of thanks could suffice.

Finally I would like to thank the staff and the other post-postgraduate students at the department for their fellowship and support.

## Table of Contents

Abstract.....	ii
Preface .....	iii
Declaration – Supervisor .....	iv
Declaration – Co-Supervisor .....	iv
Declaration – Plagiarism .....	v
Declaration – Publications .....	vi
Acknowledgements.....	vii
Table of Contents.....	viii
Acronyms .....	xii
List of Tables .....	xvi
List of Algorithms .....	xvii
Chapter 1 - Introduction .....	1
1.1 Motivation and Problem Definition .....	3
1.1.1 Problem Definition .....	3
1.2 Main Goal and Specific Objectives.....	4
1.3 Contributions .....	5
1.4 Dissertation Outline .....	5
Chapter 2 - Literature Review .....	7
2.1 Introduction .....	7
2.2 Video Tracking.....	7
2.3 Object Detection .....	10
2.3.1 Point Detection .....	10
2.3.2 Background Subtraction.....	11
2.3.3 Segmentation.....	12
2.3.4 Supervised Learning .....	13



2.4	Object Tracking .....	13
2.4.1	Point Tracking .....	14
2.4.2	Kernel Tracking.....	15
2.4.3	Silhouette Tracking .....	15
2.5	Previous Works on Video Tracking in a Maritime Environment .....	16
2.6	Conclusion.....	21
Chapter 3 - Background Modelling .....		22
3.1	Introduction .....	22
3.2	Single Image Modelling .....	22
3.3	Probabilistic Modelling .....	24
3.4	Conclusion.....	28
Chapter 4 - Level Set Methods.....		29
4.1	Introduction .....	29
4.2	Level Set Evolution .....	31
4.2.1	Reinitialisation.....	32
4.3	Classical vs. Variational Level Set Methods .....	33
4.3.1	Classical Level Set Methods .....	33
4.3.2	Variational Level Set Methods .....	33
4.4	Common Level Set Formulations for Image Segmentation .....	34
4.4.1	Local/Edge-Based Segmentation .....	34
4.4.2	Regional Segmentation .....	36
4.5	Shape Priors .....	37
4.5.1	Shape Energy Term .....	39
4.5.2	Parametric Functional.....	40
4.5.3	Multiple Shape Priors.....	42
4.6	Conclusion.....	52

Chapter 5 - Proposed Model.....	53
5.1 Introduction .....	53
5.2 Model Overview.....	53
5.3 Level Set Segmentation Algorithm.....	58
5.3.1 Tsai et al's Level Set Algorithm .....	58
5.3.2 Modifications to Tsai et al's Work .....	60
5.3.3 Implementation Details .....	61
5.4 Object Detector.....	63
5.4.1 Pre-Filter.....	64
5.4.2 Background Subtraction.....	65
5.4.3 Post-Filtering .....	69
5.4.4 Level Set Filtering .....	76
5.5 Object Tracker.....	78
5.5.1 Energy Functionals for Tracking.....	79
5.5.2 Implementation Details .....	81
5.5.3 Normalisation for Rotation/Scale Invariance.....	82
5.6 Conclusion.....	83
Chapter 6 - Experimentation and Results.....	84
6.1 Experimental Conditions.....	84
6.2 Ground Truth Estimation .....	86
6.3 Performance Metrics .....	87
6.3.1 Object Detection .....	87
6.3.2 Object Tracking .....	91
6.4 Optimisation .....	93
6.4.1 Object Detection .....	93
6.4.2 Object Tracking .....	97

6.5	Object Detection Results .....	99
6.5.1	Pre-Filter and Background Subtraction.....	99
6.5.2	Post-Filtering .....	101
6.5.3	Level Set Filtering.....	107
6.6	Object Tracking Results.....	109
6.7	Speed of Algorithm .....	116
6.8	Conclusion.....	117
Chapter 7	- Conclusion.....	118
7.1	Summary of Dissertation .....	118
7.2	Future Work .....	122
References	.....	124

## Acronyms

International Maritime Bureau	IMB
Rigid Inflatable Boat	RIB
Fast Fourier Transform	FFT
Probability Density Function	PDF
Principle Component Analysis	PCA
Singular Value Decomposition	SVD
Kernel Density Estimation	KDE
True Positive	TP
False Positive	FP
False Negative	FN
Tracker Detection Rate	TRDR
Object Tracking Error	OTE
Contour Tracking Error	CTE

## List of Figures

Figure 1-1 – Number of pirate attacks per year according to a number of sources including the IMB [3]..	1
Figure 2-1 - Overview of an object tracker [11] .....	8
Figure 2-2 - Various possible target representations for a particular object. <i>(a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette.</i> [12] .....	9
Figure 2-3 - Overlapping tiles of varying size used in [27] .....	17
Figure 2-4 - Segmentation results using [27] .....	18
Figure 2-5 - Example difference image [7] .....	18
Figure 2-6 - Example of a horizontal projection histogram (Adapted from [7]) .....	19
Figure 2-7 - Example of a horizontal projection histogram after clipping and filtering (Adapted from [7]) .....	19
Figure 2-8 - Poor contrast between ocean and target and ocean with glint [8] .....	21
Figure 3-1 - Classification as pixels as background or foreground for single Gaussian modelling. <i>If the probability of a particular pixel value is above a certain threshold, the pixel is considered part of the background</i> .....	25

Figure 3-2 - Example of kernel density estimation using single dimensional data [33]. <i>Here a normal distribution is used as a kernel</i> .....	27
Figure 4-1 - Inability of snakes to deal with changes in topology. <i>(a) is the initial contour in the image and (b) is the final contour after evolution</i> .....	30
Figure 4-2 - Example of a contour in an image (a) and the level set function that defines it (b) .....	30
Figure 4-3 - Example of a 3D level set function intersecting with a 2D plane and its ability to handle changes in topology [39].....	31
Figure 4-4 - Star-shaped interface undergoing curvature-driven evolution. <i>The tips of the star move inward, while the gaps in between them move outward</i> [42] .....	35
Figure 4-5 - Segmentation results using the Chan-Vese functional [37] .....	37
Figure 4-6 - Original image (left) corrupted by artificial occlusion and deletion (right) [44] .....	38
Figure 4-7 - Normal segmentation of corrupted image [44] .....	38
Figure 4-8 - Segmentation of corrupted image with prior knowledge of shape [44] .....	38
Figure 4-9 - Example of a shape prior encoded in function $\psi$ manipulated using $p$ to produce a level set function $\phi$ with associated contour in the image .....	41
Figure 4-10 - Segmentation with a shape prior. <i>Sub-figure (a) is the original image without removal or occlusion</i> [44] .....	42
Figure 4-11 - Segmentation with shape prior and labelling function. <i>The labelling function has values shown in sub-figure (a)</i> [44] .....	43
Figure 4-12 - Segmentation comparison of image with single shape prior vs. two competing priors [47]	45
Figure 4-13 - Segmentation of corrupted image using multiple shape priors [45] .....	47
Figure 4-14 - Example of Principle Component Analysis in two dimensions [49]. <i>The variance of the data in the original Cartesian space <math>(x, y)</math> is best captured by the basis vectors <math>v_1</math> and <math>v_2</math> in a rotated space</i> ..	48
Figure 4-15 - Example set of binary images after alignment [10].....	48
Figure 4-16 - Three dimensional example of shape model for the fighter jet training set. <i>(a) is the mean level set function, (b) is <math>+\sigma \mathbf{1} \Phi \mathbf{1}</math>, (d) is <math>-\sigma \mathbf{1} \Phi \mathbf{1}</math> and (c) and (e) are the results of their summations with the mean level set function</i> [10] .....	49
Figure 4-17 - Various samples from the training set used in Figure 4-19 [50] .....	52
Figure 4-18 - Final segmentation results for various frames of an occluded walking person segmented with a purely intensity functional [50].....	52
Figure 4-19 - Final segmentation results for various frames of an occluded walking person segmented with using a functional that includes the shape model [50].....	52

Figure 5-1 - Separate object detector and tracker architecture.....	54
Figure 5-2 - Joint object detector and tracker architecture .....	54
Figure 5-3 - Example of system functionality.....	55
Figure 5-4 - Proposed system .....	55
Figure 5-5 - Example output of the system.....	56
Figure 5-6 – Details of object detector .....	57
Figure 5-7 - Details of object tracker .....	57
Figure 5-8 - Example of a shape prior encoded in function $\psi$ manipulated using $p$ to produce a level set function $\phi$ that produces a contour in the image .....	59
Figure 5-9 - 3x3 Gaussian kernel with $\sigma^2 = 1$ .....	64
Figure 5-10 – Example frame from an image sequence .....	69
Figure 5-11 - Example raw output of background subtraction. <i>Pixels resulting from motion of objects of interest have been highlighted in red</i> .....	69
Figure 5-12 – Two-dimensional Gaussian kernel .....	70
Figure 5-13 - Example motion persistence heat-map. <i>This heat-map shows areas of high persistent motion in blue and violet, while green, yellow and red areas represent low densities</i> .....	71
Figure 5-14 - Thresholded version of Figure 5-13.....	71
Figure 5-15 – Superposition of motion image (red contours) on original frame .....	72
Figure 5-16 – Meaning of 4- or 8-adjacency. <i>For a particular pixel (shown in red) 4- adjacent pixels lie above, below, left or right (shown in green), while 8-adjacent pixels lie in all surrounding pixels</i> .....	74
Figure 5-17 - Example of tracking by comparing inner pixel models of contours. <i>As the target contour sits around the blue object in the first frame (a), the energy associated with the contour in (b) will be lower than that in (c)</i> .....	79
Figure 6-1 – Target objects as they appear in their original image sequences. <i>The sequence from which the object comes is labelled below its respective window</i> .....	85
Figure 6-2 - Table of relations between actual and predicted output.....	88
Figure 6-3 - Example image consisting of n different blobs.....	89
Figure 6-4 - Physical analogy of segmentation metrics. <i>For two intersecting regions, sub regions can be classified as contributions to one of three possible metrics</i> .....	91
Figure 6-5 – Comparison of two situations that would give a similar object tracking error. <i>Despite having similar errors, (i) would be a more desirable output.</i> .....	92
Figure 6-6 - Background subtraction results for optimal individual parameters.....	95

Figure 6-7 - Plot of ground truth area vs. vertical position in image. <i>The chosen threshold function is shown in red</i> .....	97
Figure 6-8 - Average error vs. Number of Iterations for various energy terms.....	98
Figure 6-9 - F <sub>2</sub> -Scores for various pre-filtering algorithms.....	99
Figure 6-10 - Comparison of various F <sub>2</sub> -scores for various filtering algorithms .....	102
Figure 6-11 - Output from background subtraction overlaid on original image for sequence 2.....	105
Figure 6-12 - Output from background subtraction overlaid on original image for sequence 6.....	105
Figure 6-13 - Output from background subtraction overlaid on original image for sequence 7.....	106
Figure 6-14 - Output from background subtraction overlaid on original image for sequence 10.....	106
Figure 6-15 - F <sub>2</sub> -Scores for object detector with and without level set filtering with shape priors on its output .....	109
Figure 6-16 - Comparison of target contour from sequence 4 (yellow) and the frame in which the contour started to drift from the object (red) .....	111
Figure 6-17 - Instantaneous Contour Tracking Error vs. frame number for sequence 4.....	112
Figure 6-18 - Comparison of target contour from sequence 4 (yellow) and the frame in which the contour started to drift from the object (red) .....	113
Figure 6-19 - Homogeneous nature of pixels inside object for sequence 6 and surroundings .....	113
Figure 6-20 - Instantaneous Contour Tracking Error vs. frame number for sequence 6.....	114
Figure 6-21 - Instantaneous Object Tracking Error vs. frame number for sequence 7 .....	114
Figure 6-22 - Comparison of system output at frame 88 and at frame 100 .....	115
Figure 6-23 -Instantaneous Object Tracking Error for sequence 9.....	116

## List of Tables

Table 5-1 - Statistical descriptors for pixels within a level set.....	80
Table 6-1 - Qualitative descriptions of target object size and weather for each sequence .....	86
Table 6-2 - Optimal background subtraction parameters for various sequences .....	94
Table 6-3 - Area of smallest ground truth object in each sequence.....	96
Table 6-4 - Outputs for various pre-filters .....	101
Table 6-5 - Results of filtering for various sequences and filters.....	104
Table 6-6 - P-scores for various image sequences using Chan-Vese Energy as classification criteria.....	107
Table 6-7 - Segmentation results from image sequences that passed level set filtering. <i>The boundary of the ground truth object is highlighted in green while the segmenting contour is highlighted in red.....</i>	108
Table 6-8 – Performance metrics for each sequence. <i>The sequences have been calculated using sequences of 300 frames.....</i>	110



## List of Algorithms

Algorithm 1 - Optimisation of transformation parameters for level set evolution. <i>The LSEvolution function takes input variables Im (the image), inPhi (the starting level set function), inP (the starting pose parameter vector), nlter (the number of evolution iterations) and alpha and eps vectors (step size vectors for gradient decent and gradient estimation). It returns the final level set energy, outPhi (the final level set function) and outP (the final pose parameter vector).....</i>	62
Algorithm 2 – Pseudo-code for the OptimParameter function. <i>This function serves to optimise an individual component of the parameter vector p. The function takes input variables Im (the input image), Phi (the level set function), oldP (the starting pose parameter vector), i (the number of the parameter to be optimised) and alpha and epsilon values for gradient decent and gradient estimation techniques. The output newP is a new optimised parameter vector.....</i>	63
Algorithm 3 – Pseudo-code for KDE-Based Background Subtraction. <i>The KDEBacksub function takes the current frame and the threshold from equation 5-10 as inputs and returns the background subtracted frame in outputFrame. The KDE function takes in a pixel's past values and its present value and returns a probability. The Kernel function is a Gaussian kernel as in equation 5-8 .....</i>	68
Algorithm 4 - Pseudocode for Motion Persistence Algorithm. <i>The FilterMotion function takes the input frame, a buffer of previous frames and a threshold as inputs. The threshold is used as described to determine if pixels consist of persistent motion or not. The densityFrame variable is a 2D kernel density estimate which is thresholded using the threshold input. Pixels of the input frame that are connected to the thresholded areas of threshDensity are output in outputFrame .....</i>	73
Algorithm 5 – Pseudo-code from the variable threshold connected-component filter. <i>The FilterCCVar function takes the input frame as an input. The normalised vertical position (Cy) is calculated for each connected region and evaluated using the ThreshFunc function to produce a threshold. If the region size is over the given threshold it is kept in the outputFrame .....</i>	75
Algorithm 6 – Pseudo-code for the binary image filtering using a level set with shape prior. <i>The FilterLevelSet function takes the input frame, the shape of the object sought (shapePrior) and the number of iterations (nlter) as inputs. It goes through each blob in inputFrame and returns a level set function centred on the probable location of the object in outputFrame.....</i>	78
Algorithm 7 – Pseudo-code for the level set tracking algorithm. <i>The LevelSetTrack function takes the input frame, the shape of the object tracked (shapePrior) and the number of iterations (nlter). It returns an outputFrame that is the original image with the shape drawn around the object .....</i>	81

## Chapter 1 - Introduction

While the word “pirate” brings to mind thoughts of the swashbuckling, one-eyed seafarers of childhood fantasy, the term still, unfortunately, has use in today’s modern world. Costing an estimated \$13 to \$16 billion a year [1], piracy remains a pertinent problem. Areas such as the coast of Somalia, the Strait of Malacca, Falcon Lake and recently the Gulf of Guinea are infamous for their reputation as pirate hunting grounds.

Despite increased security, piracy in these areas is increasing over the years. Inspection of pirate-related incidents per year (Figure 1-1) shows a clear increase in the number of attacks. While recent years have seen a slight drop in reported incidents of piracy, 439 attacks were reported in 2011 according to the International Maritime Bureau (IMB) [2].

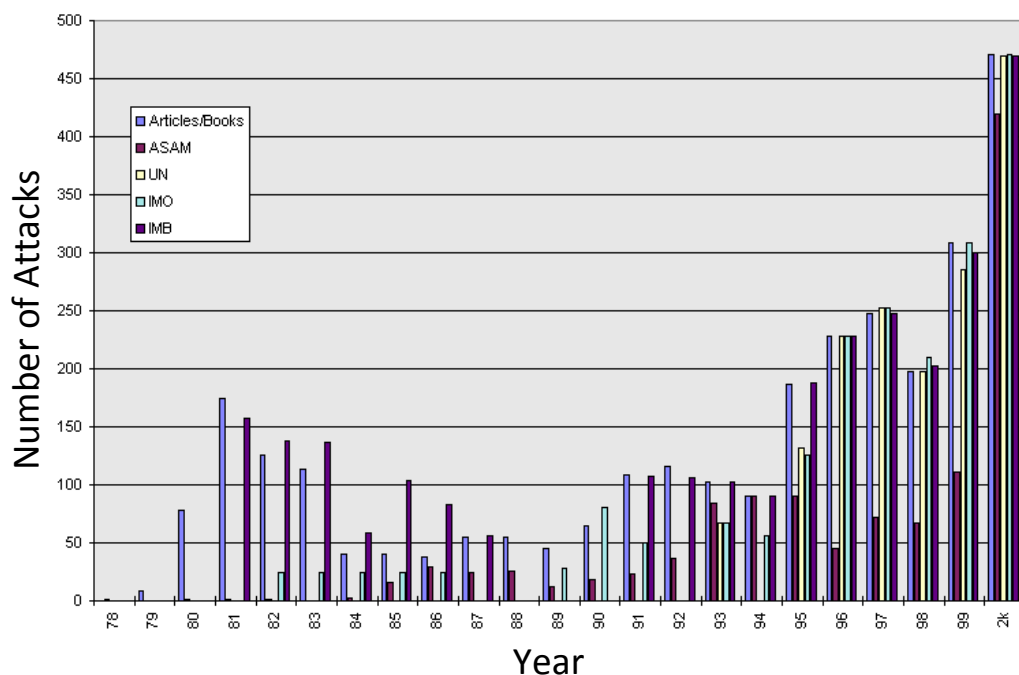


Figure 1-1 – Number of pirate attacks per year according to a number of sources including the IMB [3]

The abovementioned problem areas are subject to a high amount of international traffic, in which cargo ships must navigate through narrow bodies of water such as the Gulf of Aden near Somalia and the Strait of Malacca. This large amount of traffic requires that ships reduce their cruising speeds, making them prime targets for piracy [4]. The small motorboats that pirates use easily overtake their targets in these tight waters.

Due to the increased threat of piracy, surveillance is an absolute must on cargo ships travelling in these dangerous areas. While radar systems have been extensively used in maritime environments, these generally require large, metallic targets. Modern pirates favour small, fast Rigid Inflatable Boats (RIB) that are mainly non-metallic and thus difficult to detect [5]. While the solution to this would seem to be the use of manual detection using dedicated crew members on board, the small number at any given time makes this unfeasible.

The advent of the digital camera system has been an essential addition to maritime surveillance. Rather than requiring a number of crew members at guard points around the ship, guards may be replaced by several cameras that are monitored by a single crew member.

While this reduces the need for a large number of crew members, it still leaves a lot of room for human error. A single crew member monitoring each camera is required to stay alert at all times and identify and track various objects on the screen. This is far from ideal as human fatigue quickly sets in and after a few minutes a human observer can overlook a small vessel [6]. It is for these reasons that the surveillance process must be automated.

As humans can grow tired, there is a necessity to design and implement automated surveillance systems that are able to constantly monitor camera feeds. These video tracking systems first detect objects of interest around a ship and then track detected objects. Production of such a system is a challenging task, especially in a maritime environment:

- If cameras are to be mounted on a mast, constant motion from the ship produces a moving background. In methods that detect objects based on their motion, this would produce a large amount of errors.
- The constant motion of the sea from waves will again produce a large amount of erroneous detections using these methods, even if a camera is stationary.

A large number of previous works in literature attempt to circumvent these problems by trying to characterise the ocean. Some require the user input certain parts of the image that are known to be ocean, and attempt to characterise it according to its appearance [5,6] or group similar-looking parts of the image together in expectation that the ocean forms the largest segment [7]. These approaches then remove the “ocean” parts of the image assuming that what remains are objects of substantial interest.

While they may work well, the requirement of user input is undesirable. Furthermore, the ocean is subject to a number of changes throughout the day, and one classification that works well in the morning may not work at all by the afternoon.

Szpak and Tapamo [8] present a completely different approach that attempts to track objects using a closed curve in the image (a method known as *level set segmentation*) after they have been detected using a motion-based detection system. While the tracking results are very good, object detection suffers from detection of a large number of non-ship objects due to motion from waves.

## **1.1 Motivation and Problem Definition**

Owing to the large amount of non-ship objects that are detected in Szpak and Tapamo's work, a method of sorting actual objects of interest from the rest would be beneficial. Although a purely motion-based detector would not be able to differentiate between parts of the image belonging to the waves and those belonging to an object of interest, the difference in *shape* between the two is easy to distinguish.

This serves as a motivation for the incorporation of prior knowledge of shape into the system to increase accuracy. To do this, the use of *level set segmentation with shape priors* may be used.

A further motivation for this methodology is the lack of current object tracking literature that does use knowledge of shape to aid results in a maritime problem.

### **1.1.1 Problem Definition**

The final objective of this research is to produce a video tracking algorithm that is capable of finding and tracking various maritime objects in a given scene. The tracker should be tailored to operate in a maritime environment, which presents a unique set of image processing challenges.

Additionally, according to Dubravko *et al* [9], any maritime video tracker should:

1. "Determine potentially threatening objects within a scene containing a complex, moving background. In marine surveillance applications, it is essential that the algorithm can deal with moving background such as flickering water surfaces and moving clouds, and still detect potential objects of interest." [9]
2. "Produce no false negatives (mistakenly overlooking a ship in the image) and a minimal number of false positives (mistakenly classifying waves as a ship). A surveillance

application in general prefers no false negatives so that no potential threat is ever overlooked. On the other hand, having too many false positives would make potential postprocessing activities, such as object classification, highly impractical.” [9]

3. “Be fast and highly efficient, operating at a reasonable frame rate. The object that poses a potential threat must be detected fast so that the appropriate preventive action can be taken in a timely manner. Furthermore, if the algorithm operates at an extremely small frame rate due to its inefficiency, some potential objects of interest could be overlooked.” [9]
4. “Use a minimal number of scene-related assumptions. When designing an object detection method for marine surveillance, making the algorithm dependent upon too many assumptions regarding a scene setting would likely make the algorithm fail as soon as some of the assumptions do not hold.” [9]

The problem definition for this dissertation is thus as follows:

- Model and produce a prototype maritime video tracking system. This system will be equipped with prior-knowledge of maritime objects’ shape that may be used to aid results in both object detection and tracking. It should also be in agreement (or as much as possible) with the above requirements.
- Run the system on a set of test data in the form of maritime scenes. These scenes contain a variety of scenarios with various targets, viewpoints and weather conditions.

## **1.2 Main Goal and Specific Objectives**

The main goal of this work is to find a way to introduce knowledge of shape into the maritime tracking process that allows it to increase segmentation accuracy as much as possible. Specifically, the objectives that need to be achieved in this work are to:

- Investigate and present current methodology for the incorporation of prior knowledge of an object’s shape into segmentation using level sets.
- Design and implement a video tracking system that must produce object detection results that ideally have zero false negative and have as few false positives as possible, using shape knowledge.

- Use this knowledge for object tracking, maximising the number of frames for which the algorithm is able to track the object while maintaining a correct boundary around the object at all times.

### 1.3 Contributions

This work contributes in the following ways:

- Design of a model that incorporates shape knowledge into object detection and tracking. This is in the form of a modification to the method presented by Tsai *et al* [10] to incorporate shape knowledge into level set segmentation.
- The later modification opens up a number of possibilities for alterations of the algorithm's behaviour. These include modifications for better segmentation results in binary images, and incorporation of both shape and interior pixel models for object tracking.

### 1.4 Dissertation Outline

Chapter 2 introduces the concept of video tracking and covers introductory theory to the subject. The subject of video tracking is a broad one containing a number of different approaches to the problem. This chapter tries to group these approaches into different varieties and introduces elementary theory on each variety of both object detection and object tracking. Finally, the various attempts at applying these methods to a maritime tracking problem are discussed.

As Szpak and Tapamo's approach [8] is used as a foundation for the work presented in this dissertation, various methods of applying background subtraction (the object detector used in [8]) are reviewed in Chapter 3 followed by a review of level set segmentation methods (the object tracker used) in Chapter 4.

Chapter 4 contains a full discussion on the introduction of shape knowledge into the use of level sets for segmentation, including the various ways it is achieved and how multiple shapes may be incorporated.

Chapter 5 presents explicit details of the final model implemented. The overall system is presented, followed by a discussion of the level set algorithm used. This algorithm is presented in detail, followed by novel modifications to it. The chapter continues with details of the various sub-systems of the object detector and object tracker. It is further shown how the background subtraction algorithm is implemented, how the results are filtered and finally classified using a

level set method with shape priors. These modules form the basis of the object detection scheme. This is followed by a discussion of the method for tracking objects once they have been found in the scene using a level set method with shape priors.

Chapter 6 presents results achieved. Metrics used to measure data and their meanings are covered, followed by a discussion of how various parameters in the system are optimised. Finally, results of the object detector and object tracker are presented and discussed.

Chapter 7 summarizes the main elements of each chapter and outlines some ideas for future work.

## **Chapter 2 - Literature Review**

### **2.1 Introduction**

It has been shown that the maritime environment has a clear need for automated object detection and tracking methods for security purposes, particularly in the unexplored field of level set segmentation using shape prior information. This chapter explores the components of such an automated object detection and tracking system and the variety of methods that are used. This includes details on object detection and object tracking, and a discussion of previous work on video tracking in the maritime environment.

### **2.2 Video Tracking**

Video tracking is loosely defined as the “process of estimating over time the location of one or more objects using a camera” [11] and systems generally consist of an object detector and object tracker. Video tracking is used in a variety of tasks such as [12]:

- Human identification based on motion.
- Automated surveillance.
- Video indexing: Automated annotation and retrieval of videos in multimedia databases.
- Human-computer interaction including iris tracking and gesture recognition such as Microsoft’s Kinect device [13].
- Traffic monitoring that provides real time statistics of traffic flow.
- Vehicle navigation that uses video-based path planning and obstacle avoidance.

Due to the unpredictable nature of video input, there are many challenges in the design of a video tracker. These include:

- Changes in pose of a target - If a tracker is designed to detect objects that are in a particular orientation (say cars from the side) changes in the objects pose (the car turns and faces the camera) may cause errors.
- Changes in ambient illumination due to weather conditions in an outdoor scene. This is particularly pertinent in a maritime environment where the sun may cause excessive amounts of glint on the ocean surface at particular times of the day.
- Noise from the camera.



- Occlusions from a target moving behind an object, obscuring its view. Thankfully in a maritime environment this is seldom a problem as objects are so spaced out from one another.

Certain assumptions, however, allow for simplification of the video tracking system. Some examples of common assumptions are:

- The camera is stationary (no camera panning).
- The motion of objects is smooth.
- Objects move with constant velocity or acceleration.

While the design of a video tracker varies from one method to another, Maggio and Cavallaro [11] suggest a generalisation of the system as shown in Figure 2-1.

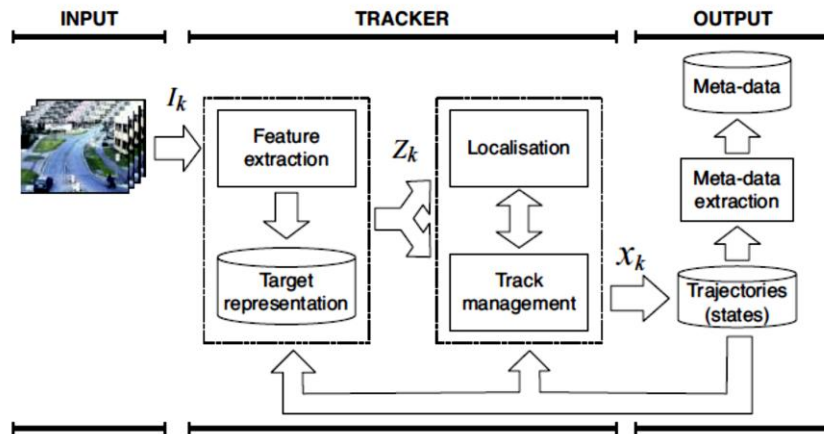
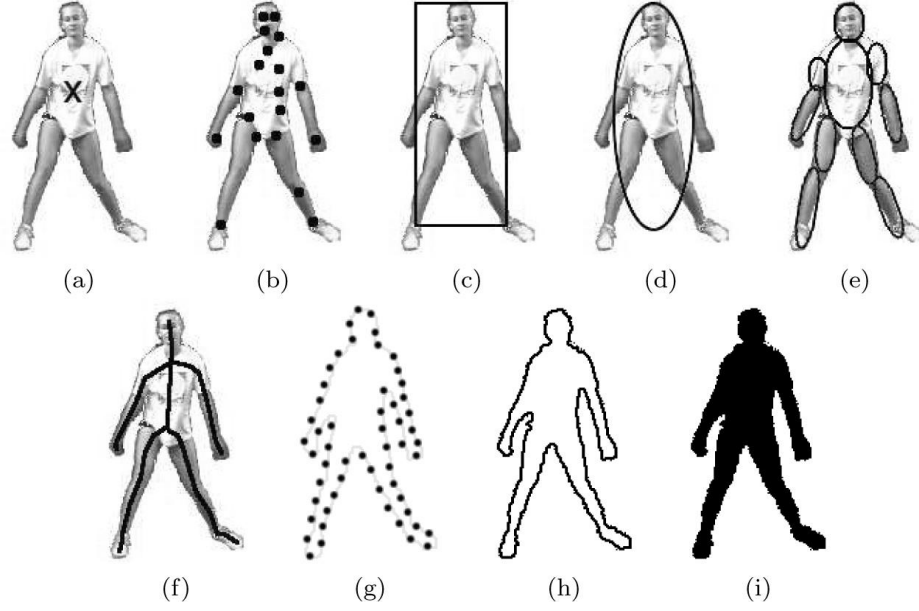


Figure 2-1 - Overview of an object tracker [11]

The sequence  $I = \{I_k : k \in \mathbb{N}\}$  represents the input frames of the video sequence. Each frame  $I_k$  is an element of  $E_I$  which is defined as the space of all possible images. For a single object, object tracking produces an estimation of the time series  $x = \{x_k : k \in \mathbb{N}\}$  where  $x_k$  represents the state of the target for frame  $k$ . This state is dependent on the target representation, which is dependent on the algorithm used. Figure 2-2 shows a number of target representations using a human as a target object.



**Figure 2-2 - Various possible target representations for a particular object.** (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette. [12]

In order to estimate target states,  $I_k$  may be mapped from the image space  $E_I$  to the feature space  $E_o$  which highlights information that is relevant to tracking. This method is known as *feature extraction* and produces a set of observations in feature space  $z_k \in E_o$ .

From the extracted features, the location of the object is *detected*. It is necessary to detect the object in either every frame of the input video sequence or at least the first frame that it appears on.

Once an object has been found, it is necessary to *track* its trajectory over time by finding correspondences in object position from frame to frame. Trackers are generally separated into those that perform object detection in multiple frames by an object detector described above and then try to find correspondences between them, or those that perform object detection and correspondence at the same time. The latter case initially detects the object and then updates its location from frame to frame.

There are various methods of both object detection and tracking which are largely dependent on the states that are to be detected. Various object detection and object tracking methods are reviewed next.

## 2.3 Object Detection

Object detection is necessary in order to find a target either in the first frame in which it appears or every single frame that is present. Commonly, each frame is analysed individually, however some algorithms make use of temporal information that is computed from a number of frames of the sequence [12].

Although there are a large number of different methods available, object detection can be categorised into four different varieties:

- **Point detectors**, which are used to find a number of single interest points in an image.
- **Background subtraction**, which builds a model of the background of a scene and considers deviations from it to be objects of interest.
- **Segmentation**, which partitions an image into groups of similar regions.
- **Supervised learning**, which automatically learns the views of a particular object to find it within a scene.

These methods are next explained in detail.

### 2.3.1 Point Detection

This method of object detection involves finding a set of interest points in an image. The fundamental assumption is that the object being detected will contain a set of points that is unique. By looking for this pattern in the entire image, the object can be found. A substantial advantage is that detected points are generally invariant to changes in illumination and camera viewpoint.

A rudimentary point detector was first introduced by Moravec [14]. This corner detector moves a local window through the image and calculates the change in its average intensity when shifting it in various directions from a particular point. Small changes in all directions indicate a flat region; large changes in only two opposing directions indicate an edge, while large changes in all directions are considered corners.

The Harris detector [15], builds on the above technique by calculating first order image derivatives in both horizontal  $I_x$  and vertical  $I_y$  directions, and calculating the second moment of a small neighbourhood of pixels:

$$M = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \quad 2-1$$

The variation of this neighbourhood is measured by calculating  $R$ :

$$R = \det(M) - k * \text{tr}(M) \quad 2-2$$

Where  $\det()$  and  $\text{tr}()$  are the determinant and trace functions respectively.  $R$  is then thresholded to determine if the neighbourhood has a feature point in it. The main benefit of using the Harris detector is the invariance of the matrix  $M$  to rotation and translation.

It is, however, not invariant to changes in scale and so Lowe [16] introduces the Scale Invariant Feature Transform (SIFT) to solve this problem. In order to be scale invariant, the algorithm is required to find features that are “stable” across all possible scales. To do this, it uses a continuous function of scale known as scale space, which is produced by convolving an image with a variable-scale Gaussian function to simulate the effect of down-scaling. This produces a family of images that are separated by a factor that is an argument of the Gaussian function. Finding a feature present in the same location across all of these images would indicate its invariance to scale changes.

### 2.3.2 Background Subtraction

A second common method of object detection is background subtraction, which removes regions of an image deemed not to be of interest. If an algorithm is able to model the background of an image, any deviation from this can be identified as a possible object of interest. This of course requires that both a number of frames from a video sequence are available to build the model and

that objects to be detected are moving. If objects are stationary, they become part of the background model and are not detected.

As it is used in the final model, this type of algorithm is covered in explicit detail in Chapter 3.

### 2.3.3 Segmentation

Segmentation is the subdivision of a particular image into its consistent regions or objects [17]. Ideally segmentation should stop when objects of interest (for a particular application) are isolated. In an object detection process for video tracking, segmentation algorithms seek to partition the image into regions belonging to objects of interest, and the background.

Segmentation may be treated as a clustering problem. Here each pixel is represented by a set of features  $F_p = \{f_1, f_2, \dots, f_n\}$  calculated from it that may be further represented as a point in  $n$ -dimensional feature-space. High-density regions of pixels in this space are considered pixels of a single region with some specific properties and are clustered together. Clustering is a non-trivial process and thus there are many different ways of performing it.

Comaniciu and Meer [18] present a method that clusters pixels using the Mean-Shift algorithm. The features used for each pixel consist of the pixel's colour vector and its image coordinates. Pixels that are clustered together in this space will then be those that are similar in colour and position. This ideally reduces an image to a set of connected regions (or blobs), each homogeneous in colour.

Mean-Shift is a method of clustering where a number of cluster centres are randomly chosen within a feature-space. For each cluster centre, an ellipsoid centred on this point is drawn and the mean of all feature-points lying within it is calculated. The cluster centre is then moved towards this mean point. This process is repeated iteratively until each cluster centre does not move.

Shi and Malik [19] treat segmentation as graph-partitioning problem. Here, each pixel is considered a vertex of a connected graph. By pruning the joints or "edges" between graph vertices, one expects to produce a number of sub-graphs each representing the segmented image regions. Each edge may be weighted and in this case the total weight of edges pruned is called a "cut". The weight of an edge may be calculated from a number of factors including the colour, brightness or texture similarity between the vertices it links. In order to group similar regions together, it is intuitive to make cuts that have a minimum value (i.e. separate dissimilar regions),

however this often results in over-segmentation in methods that use this [20]. Shi and Malik rather calculate a normalised version of the cut, the minimum of which produces better segmentation results.

Segmentation may also be achieved using *active contours*. Here a closed contour, which is used to separate regions in the image, is iteratively evolved in the image. As it is used in the final system model, details of this methodology, specifically a derivative thereof known as *level set methods* are discussed in Chapter 4.

#### **2.3.4 Supervised Learning**

Object detection can be implemented by supervised learning, where an intelligent agent is taught different views of the object in question from a set of learning examples.

Viola *et al* [21] present a pedestrian detection system that operates using this principle where a classifier is designed to perform binary classification on small images as being a pedestrian or not. A number of filters are applied to frames where each filter is designed to measure a particular feature. A classifier is built as a weighted sum of the filter outputs which is thresholded to determine presence of a pedestrian. To train the agent, a training set of images is applied to the input of the classifier. Filter weights and the threshold are adjusted to lower the error between the classifier's output and the expected output for the images.

### **2.4 Object Tracking**

The purpose of an object tracker is to find correspondences between consecutive image frames, in other words to match up objects in one frame to objects in the next. The function of the tracker is largely dependent on the type of object detection used: The function of the tracker is largely dependent on the type of object detection used. Again, trackers are grouped into different varieties:

- **Point trackers**, which match feature points detected in each frame.
- **Kernel trackers**, which are designed to track a pixel region within a geometric shape.
- **Silhouette trackers**, which are able to track arbitrary object outlines.

Details of the above methods are described below.

### 2.4.1 Point Tracking

In the context of point tracking, the correspondence problem refers to matching identifiable feature points found using point detection algorithms in a particular frame with those in the next frame. A correspondence of a particular point in a frame refers to that point's location in a subsequent frame.

Salari and Sethi [22] introduce a method of addressing two fundamental problems that face this type of tracking, specifically occlusion and poor feature point detection. It is assumed that one has  $n$  frames in a sequence with  $m$  feature points in every frame. Assuming that these points belong to real world objects, their trajectories are assumed to be smooth. It is intuitive that the correspondence for each point in a particular frame should be initialised to its nearest neighbour in the consecutive frame. A *greedy exchange algorithm* would then exchange correspondences between points to minimise some cost. However this method fails if actual correspondences are occluded or if they have not been picked up due to poor detection. The authors solve this problem by introducing phantom points that are used as fillers to extend trajectories. Points whose nearest neighbour correspondences are outside a threshold distance are considered incomplete trajectories and phantom points added to them. For every trajectory an additional trajectory is created solely from phantom points and points are exchanged between them to maximise a gain. This gain is a function of how smooth the trajectory is and whether points are phantom or not, naturally penalising them if they are.

Broida and Chellappa [23] model points found in images as corners of a rigid body undergoing rotational or translational motion. The body is assumed to be undergoing constant motion, being modelled as an unknown centre of mass moving with constant but unknown translational velocity. Points form moment arms with the centre of mass whose unknown phase angles change at an unknown rate. Tracking is achieved by estimating these unknown object motion parameters using detected feature points. This can be done in two ways: *Recursive solutions* look at points frame by frame successively improving estimates as more points are used or *batch solutions* look at points from every frame and estimate parameters according to an objective function. The authors opt for a recursive solution using a Kalman filter. A Kalman filter uses noisy measurements observed over time and outputs a "truer" value of these measurements by calculating a predicted value, and outputting a weighted average of this and the original depending on a calculated uncertainty of the predicted value.

### 2.4.2 Kernel Tracking

Rather than using individual points on an object for tracking, a kernel can be used. In an object-tracking context the concept of a kernel refers to the shape and appearance of an object. For example, the kernel can be a rectangular template or an elliptical shape with an associated histogram [12].

Comaniciu *et al* [24] use a kernel consisting of an elliptical region surrounding the target combined with the colour histogram of the pixels within it. For frames where the target location is unknown, a candidate kernel can be created from any arbitrary elliptical region within the image. The candidate and target histograms are compared using a similarity function. Tracking is achieved by altering the position of the candidate elliptical region to maximise similarity.

The similarity function has large variations for adjacent locations in the image which reduces the smoothness of the similarity function. As a consequence, estimating the direction of increasing similarity becomes problematic and requires exhaustive searches in the image every frame. The authors' solution to this was to mask the elliptical region with an isotropic kernel so that pixels would have smaller weights (and thus play a smaller role in the histogram) the further from the centre they are. This ensures a smooth similarity function, and the search for correspondences can be confined to a local region around the last known location of the target.

### 2.4.3 Silhouette Tracking

Silhouette tracking is used when simple geometric shapes (such as ellipses) are not sufficient to describe objects with complicated boundaries (such as a person's hands or head and shoulders). The goal is to find an object region in each frame using an object model generated from previous frames [12].

Huttenlocher *et al* [25] achieve this using shape matching methods. The motion of the silhouette is decomposed into two-dimensional motion and two-dimensional shape change components. Under the assumption that the shape change is gradual from frame to frame, the motion is easily factored out and the shape change can be represented as a sequence of 2D models.

Bertalmio *et al* [26] present a method that uses level set segmentation for tracking. Level set segmentation is introduced as a derivative of active contour methods in section 2.3.3. While the finer details of the method are discussed later in Chapter 4, this method surrounds the tracked object with a contour. The final contour from a previous frame is used as an initial contour in the



next. This increases efficiency by using information from a previously segmented frame in the current one.

## **2.5 Previous Works on Video Tracking in a Maritime Environment**

The following discussion is a review of previous attempts at introducing object tracking to the maritime field. As it has been mentioned earlier, the maritime environment is a particularly challenging one for tracking and thus the majority of works discussed here deal with very low-level solutions to the problem.

As the ocean, constantly filled with moving waves, is prone to producing erroneous detection with methods that detect moving objects, some authors choose to characterise it and label pixels that don't match this characterisation as objects of interest.

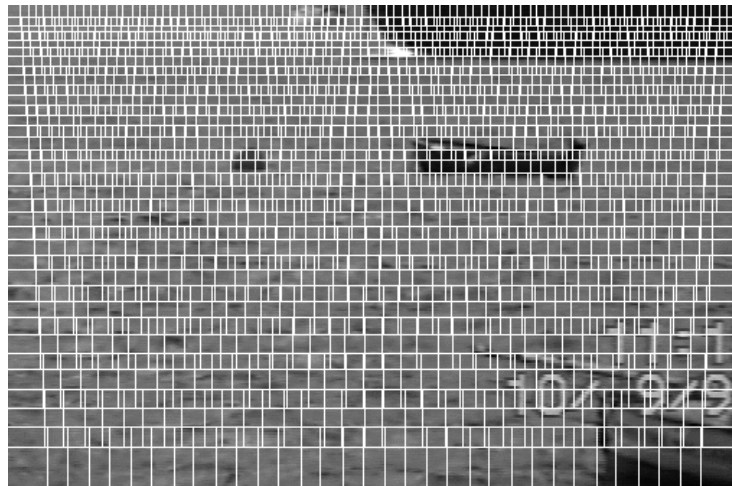
Sanderson *et al* [5] implement an algorithm that does this using frequency information. The algorithm's initial stage selects 10 points that are known to be part of the sea and applies a 32 by 32 pixel window to each. The Fast Fourier Transform (FFT) of each window is calculated and is used to model the current sea state with a characteristic set of frequencies. This step constitutes the calibration phase of the system.

Once these frequencies have been found, difference images are created for each input frame by transforming the frame into the frequency domain, subtracting the sea frequencies and then transforming this back into the spatial domain. This constitutes detection of objects of interest. The result passed onto the object tracking algorithm that uses a frame differencing technique to detect motion cues in the difference images. This technique works by statistically analysing four by four pixel regions over a number of frames. Regions that change statistically constitute ones that contain motion, while those that remain the same are labelled as static. Detected objects are then tested against a set of motion constraints. These constraints are known prior to tracking and enforce certain limits on maximum acceleration and orientation change of objects of interest. Objects that do not satisfy these constraints are ignored.

Smith and Teal [6] implement a similar approach using a histogram-based descriptor of the appearance of the sea. Rather than characterising the sea by its frequency, the algorithm learns its grey level distribution in a 32 by 32 pixel tile. The image is subsequently divided into overlapping

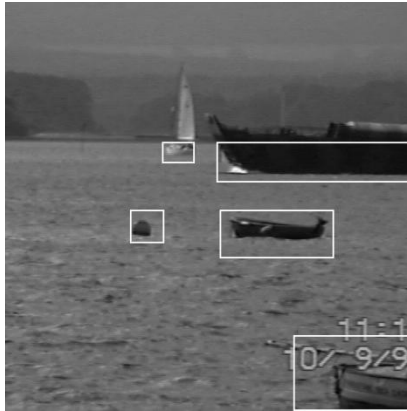
32 by 32 pixel tiles and those tiles whose distribution match that of the sea are labelled as such. Thereafter, tracking is implemented in the same manner as in [5].

Voles and Teal [27] continue with the theme of using crude descriptors of tiles in an image. Differences between their method and earlier papers include the use of a varying tile size that overlap as shown in Figure 2-3. The thinking behind this is that the ocean is a horizontal plane, and thus objects closer to the camera will appear larger compared to those further away and therefore require larger tiles. Due to this reason, tiles increase in size towards the bottom of the image. The tiles overlap so that the algorithm is able to pick up smaller objects that may be only partially covered by a segment.



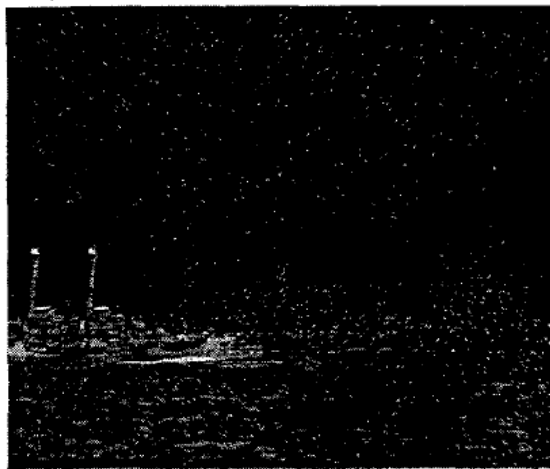
**Figure 2-3 - Overlapping tiles of varying size used in [27]**

For each tile a set of four features are calculated that constitute a feature vector. Similarly to the clustering method in [18], these can be considered as points in a four dimensional feature-space. The underlying assumption is that a large main cluster of points would belong to tiles that lie on the sea, while outliers would belong to tiles with objects of interest in them. The separation (or clustering) of these two groups of points can be done using a number of methods ranging from using neural networks to purely statistical methods. The major limitation of this method is its imprecise segmentation results, for example those shown in Figure 2-4. Bounding boxes are at times far larger than the target they contain, and are clipped above the horizon.

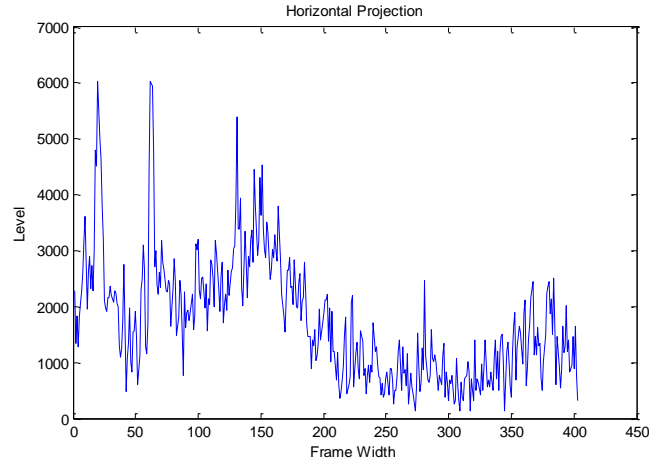


**Figure 2-4 - Segmentation results using [27]**

Voles *et al* [7] combine the use of these same four features from Vole and Teal [27] with motion information obtained by frame differencing. A simple difference image is created by subtracting the current image's pixel values from those of the previous and taking the absolute value of the result. Regions containing more motion will have brighter pixel values in this image. The image is projected onto horizontal and vertical axis that results in two projection histograms. An example of a difference image and its associated horizontal projection histogram are shown in Figure 2-5 and Figure 2-6 respectively.

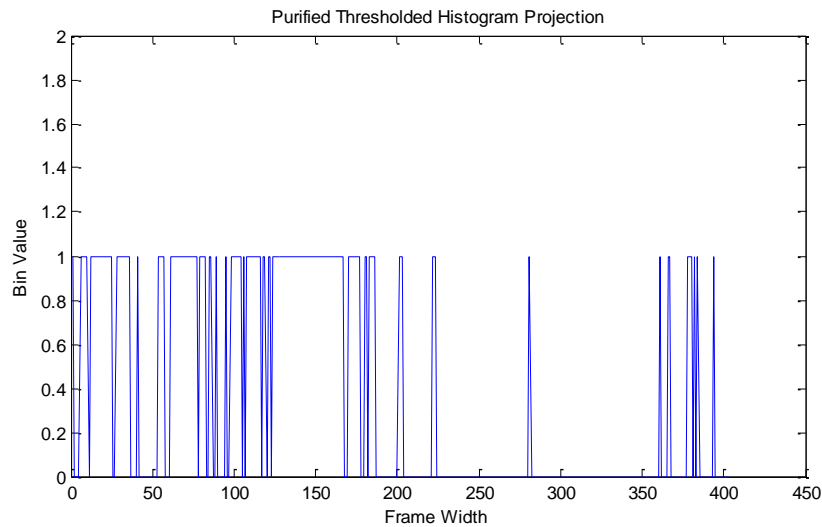


**Figure 2-5 - Example difference image [7]**



**Figure 2-6 - Example of a horizontal projection histogram (Adapted from [7])**

The histograms are filtered to remove small incidents of motion due to the sea and finally clipped so that regions above a certain threshold are set to one and below it to zero. An example of the result of this is shown in Figure 2-7.



**Figure 2-7 - Example of a horizontal projection histogram after clipping and filtering (Adapted from [7])**

Recombining filtering and clipped projection histograms produces a binary map where moving items are represented by white rectangles and stationary background is black.

The next stage splits the image into tiles and calculates a four dimensional feature vector for each as implemented in [27]. Rather than clustering points in four-dimensional feature space, a Euclidian distance map is created by calculating the Euclidean distance between the feature vector

of a tile in the current frame and its vector in the previous one. Naturally, areas of large Euclidean distance are those that contain motion. Combining this with the previously calculated binary map allows the algorithm to calculate areas that probably contain motion of a vessel. The algorithm is purely pixel-based and therefore fails to segment larger maritime objects.

Socek *et al* [9] present a method of combining existing object detection methods with colour information. It initially segments the image with background subtraction using a Bayes decision framework which works best for backgrounds with complex variations and that are not periodic. In a maritime environment, the algorithm suffers from poor segmentation results having inaccurate boundaries and many scattered pixels. The authors seek to solve these issues by combining results with that of colour-based segmentation. The colour segmentation is treated as a graph-partitioning problem (See section 2.3.3 (Segmentation methods) above for details) and combining it with background subtracted output results in enhanced performance.

Szpak and Tapamo [8] introduce an approach that uses a video tracking approach quite different from that of the above algorithms. Object detection is implemented using a modified method of single Gaussian background subtraction, the details of which can be found in Chapter 3. Where normal background subtraction deals with pixels in isolation, Szpak and Tapamo enforce a spatial-smoothness constraint that deals with neighbourhoods of pixels. The constraint assumes that real-world objects are spatially consistent entities and requires that a whole group of pixels, rather than single ones, exhibit motion behaviour before marking them as such.

The output of this method is further segmented using level set methods, a derivative of active contour-based segmentation. This contour is used in the object tracking phase as described in by Bertalmio *et al* in [26], where the final contour from the previous frame is used as an initial contour in the next.

The algorithm was tested on 17 test sequences. The algorithm was able to successfully track in all but three of the given sequences. The algorithm even showed good results in overcast and rainy conditions. It failed in sequences where there was insufficient contrast between the ocean and the target, and thus fails to pick up specific motion of the target; when the target moves too slowly and is thus considered part of the background and when there is a lot of glint in the scene. Figure 2-8 is an example of poor contrast between ocean and target and large amounts of ocean glint.



**Figure 2-8 - Poor contrast between ocean and target and ocean with glint [8]**

Due to its high success rate and possible avenues for improvement it was decided to base further work on the model proposed by Szpak and Tapamo [8]. As this is an application of both background subtraction and level set methods, Chapter 3 and Chapter 4 include a full discussion of the specifics of each technique.

## **2.6 Conclusion**

This chapter discusses introductory theory to the variety of methods used in video tracking. The concept of a video tracker and its constituent object detector and object tracker models and some general theory related to the subject are introduced. Various categories of object detectors are reviewed, followed by a discussion of the various methods used to track detected objects through the video sequence. Finally, as this is the area in which this work is concerned, various previous techniques for applying video tracking to a maritime environment are explored.

## Chapter 3 - Background Modelling

### 3.1 Introduction

Various methods applying video tracking to a maritime environment have been introduced. An emphasis has been placed on Szpak and Tapamo's work [8] which uses background modelling as a method to detect objects of interest in a scene. As their model forms a basis for this work, several well-known techniques for background modelling are now discussed.

Background modelling is a method of object detection that is achieved by building a representation or model of the background of a particular scene and finding deviations from this model [12].

Background modelling is reliant on two basic assumptions. Firstly it assumes a number of image frames  $I_k = \{I_1, I_2, \dots, I_t\}$  are available in the form of a video sequence of the scene. Secondly it assumes that objects to be detected in this sequence are moving in some way around the scene.

By creating a model of the background of the scene from a number of frames of the input sequence, moving or foreground objects can be detected by finding deviations from this model in successive frames. This has a number of challenges as the background is itself subject to changes from illumination and non-stationary background objects such as swinging leaves, rain, snow and shadows caused by moving objects [28]. While the techniques presented in this chapter are designed for grayscale images, they may be extended to colour images if required.

### 3.2 Single Image Modelling

There are a number of rudimentary methods that can be used for background modelling, each with their own drawbacks. These methods use a model of the background that is a *single image*, the same size as a single frame. Background subtraction for this method can be generalised as follows: Assuming the current background model for the  $t^{th}$  frame  $B_t$ , the output of the background subtraction algorithm for the current frame  $BS_t$  at pixel  $(x, y)$  can be calculated as:

$$BS_t(x, y) = \begin{cases} 1 & \text{if } |I_t(x, y) - B_t(x, y)| > Th \\ 0 & \text{otherwise} \end{cases} \quad 3-1$$

where  $Th$  is a predetermined threshold. If  $BS_t(x, y)$  is equal to 1, this pixel is considered to be part of a foreground object, while it is considered to be part of the background if it is 0. This essentially thresholds the difference between the current and the background estimate. While this form of subtraction is fairly standard amongst algorithms, the method of actually modelling the background is what differs and is discussed further.

The simplest form of background modelling is known as *frame differencing* and using the previous frame as the background model for the next:

$$B_t = I_{t-1} \quad 3-2$$

While this method is easy to implement due to its simplicity it is very sensitive to the threshold  $Th$  and highly dependent on an object's speed and the camera's frame rate.

More information may be incorporated into the model by using an average of the previous  $n$  frames; however each of these frames must be stored in memory. This can be solved by using a running average:

$$B_t = \alpha * I_{t-1} + (1 - \alpha) * B_{t-1} \quad 3-3$$

Where  $\alpha > 0$  is a learning rate. Here the current background estimate is calculated as the weighted sum of the previous background estimate and previous input frame. Selectivity can be used to improve this model by only updating the previous background for a particular pixel if it is marked as the background in previous frames:

$$B_t(x, y) = \begin{cases} \alpha * I_{t-1}(x, y) + (1 - \alpha) * B_{t-1}(x, y) & \text{if } BS_{t-1}(x, y) = 0 \\ B_{t-1}(x, y) & \text{if } BS_{t-1}(x, y) = 1 \end{cases} \quad 3-4$$

The background may also be estimated to be the median of the intensity of the pixel over time at each pixel [29]. Here a buffer of previous frames is used and the assumption is that the pixel stays in the background for more than half of the buffered frames. Rather than using a buffer, the median may also be recursively estimated as proposed by McFarlane and Schofield [30].



### 3.3 Probabilistic Modelling

The problem with using a single image as a background model is that each pixel in the current frame has only a single value to compare itself to. For a background that may have some cyclic information in it, for example a pixel oscillating between two values, using the mean or median of the cycling values may incorrectly model a particular pixel.

Probabilistic methods model the probability of a pixel having a particular value in the current frame. Each pixel is assigned its own probability density function (PDF). To do this for a pixel at  $(x, y)$  a distribution  $f_{t,(x,y)}$  is modelled using previous values, where  $f_{t,(x,y)}(p)$  is the probability of the pixel having a particular intensity  $p$ . A pixel from the current frame is marked as the foreground if its value is seen as *unlikely* by the distribution. More formally:

$$BS_t(x, y) = \begin{cases} 1 & \text{if } f_{t,(x,y)}(I_t(x, y)) > Th \\ 0 & \text{otherwise} \end{cases} \quad 3-5$$

As all of the following methods deal with pixel-specific probability distributions, the notation  $BS_t(x, y)$  shall be shortened to  $BS_t$  for brevity. Unless otherwise stated, all variables discussed belong to the same pixel at some arbitrary image coordinates.

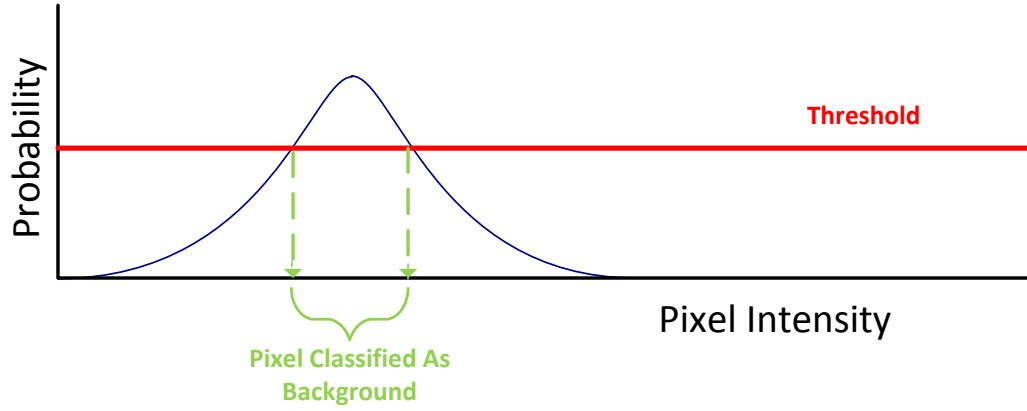
The concept of probabilistic modelling is best illustrated using the following example. Wren *et al* [34] propose the use of a single Gaussian to model the distribution of each pixel. Here the distribution for a pixel in the  $t^{th}$  frame is modelled as:

$$f_t(p) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{(p-\mu_t)^2}{2\sigma_t^2}} \quad 3-6$$

Where the current mean  $\mu_t$  and variance  $\sigma_t$  of the distribution are iteratively updated for each frame as:

$$\begin{aligned} \mu_t &= \alpha I_t + (1 - \alpha) \mu_{t-1} \\ \sigma_t^2 &= \alpha (I_t - \mu_t)^2 + (1 - \alpha) \sigma_{t-1}^2 \end{aligned} \quad 3-7$$

Figure 3-1 shows a possible distribution for a particular pixel. For a certain threshold, the brackets show a *range* of possible intensities that may be considered background values. If the pixel's next value falls within this range, it is considered part of the background, otherwise it is considered part of the foreground.



**Figure 3-1 - Classification as pixels as background or foreground for single Gaussian modelling.** *If the probability of a particular pixel value is above a certain threshold, the pixel is considered part of the background*

The problem with single Gaussian methods is that while they are able to model variations of a pixel around a particular value (as above in Figure 3-1), they are unable to model distributions that may be multi-modal (one that has multiple peaks). Stauffer and Grimson [31] present a method of dealing with multimodal distributions by modelling each pixel with a *mixture* of  $K$  Gaussians:

$$f(p) = \sum_{i=1}^K \omega_{i,t} * \eta(p, \mu_{i,t}, \sigma_{i,t}) \quad 3-8$$

Where  $\eta$  is a Gaussian PDF with mean  $\mu_{i,t}$ ,  $\sigma_{i,t}$  weighted by  $\omega_{i,t}$  evaluated at  $p$ .

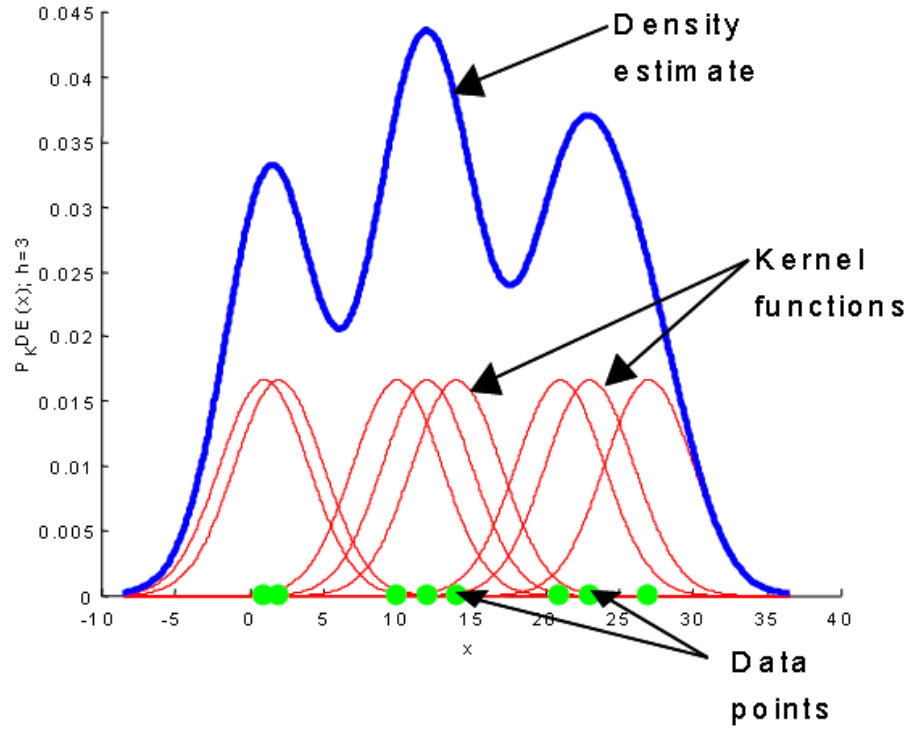
The following process is followed for each new pixel value  $I_t$ :

- The Gaussians are ordered in decreasing order of  $\omega/\sigma$ , a value that increases as the distribution gains more evidence and variance decreases. The top distributions are the most probable to be those that model the background.
- The value is checked against each of the  $K$  distributions, where a match is a value within  $2.5\sigma$  of the distribution.
- If none of the distributions match, the least probable distribution is replaced by a new one, with mean equal to  $I_t$  and an initial high variance and low weight.
- The weight of each distribution is adjusted as follows:  $\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha M_{k,t}$  where  $\alpha$  is the learning rate and  $M_{k,t}$  is one if there is a match and zero if not. The weights are then normalized.

- The parameters  $\mu$  and  $\sigma$  of unmatched distributions stay the same. Matched distributions are updated as follows:  $\mu_t = (1 - \rho)\mu_{t-1} + \rho I_t$  and  $\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(I_t - \mu_t)^2$  where  $\rho = \alpha\eta(I_t, \mu_{i,t}, \sigma_{i,t})$  is a second learning rate.
- In determining if a pixel is foreground or background:
  - The first  $B$  distributions are chosen as the background model where  $B = \text{argmin}_b(\sum_{k=1}^b w_k > T)$  and  $T$  is a measure of the minimum portion of data that should be accounted for by the background.
  - Pixels that do not match any of these first  $B$  distributions are deemed to be part of the foreground.

While it is clear that this method is able to handle multimodal distributions it has an added advantage that if a pixel value is allowed to become part of the background, it doesn't destroy the existing model. Only once a distribution reaches the  $Kth$  most probable on the above list and a new pixel value is observed is it allowed to be removed from the model.

Elgammal *et al* [32] introduce the kernel density estimator to model the distribution. Kernel density estimation (KDE) is a method of estimating an unknown distribution given some independent samples drawn from it. A symmetric function that integrates to one (known as a kernel) is placed at each sample. A particular value's probability can be calculated as the sum of all kernels at that point. Figure 3-2 shows an example of kernel density estimation using some single-dimensional data points using a Normal Distribution as a kernel.



**Figure 3-2 - Example of kernel density estimation using single dimensional data [33]. Here a normal distribution is used as a kernel**

For some samples  $\{x_1, x_2, \dots, x_n\}$  an estimation of their distribution is:

$$\hat{p}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \quad 3-9$$

Where  $K$  is the kernel and  $h > 0$  is the bandwidth of the kernel. The bandwidth is a free parameter that controls the width of the kernel and must be chosen carefully. In a background modelling context this process can be applied to the previous  $L$  pixel values  $\{I_{t-L}, I_{t-L+1}, \dots, I_{t-1}\}$  such that:

$$f(p) = \frac{1}{Lh} \sum_{i=t-L}^{t-1} K\left(\frac{p - I_i}{L}\right) \quad 3-10$$

There are a number of different possible kernels that may be used; however the Normal Distribution is the most documented and does not require the user to arbitrarily define bandwidth values. While the kernel density estimator is resource intensive in that it, unlike Gaussian

modelling, requires a buffer of the  $L$  previous frames, the major benefit of using it is that it is able to model each pixel distribution very accurately. It has been shown [34] that for an infinite number of observations kernel density estimators converge to the distribution that they were drawn from.

### **3.4 Conclusion**

This chapter introduces and details the use of background modelling for detection of objects in a video tracking system. It covers two common varieties of background models: those that create a model that consists of an image of the estimated background, and those that create a probability model for each pixel. Detection is achieved by comparing pixel values from the current frame with the pixel values of the background image in the first case or by calculating and thresholding their probability in the second.

## Chapter 4 - Level Set Methods

### 4.1 Introduction

The concept of a video tracker and its application to a maritime environment has been introduced. As Szpak and Tapamo [8] use an application of both background subtraction and level set methods, Chapter 3 has covered background subtraction in detail. The concept of level set segmentation is now introduced.

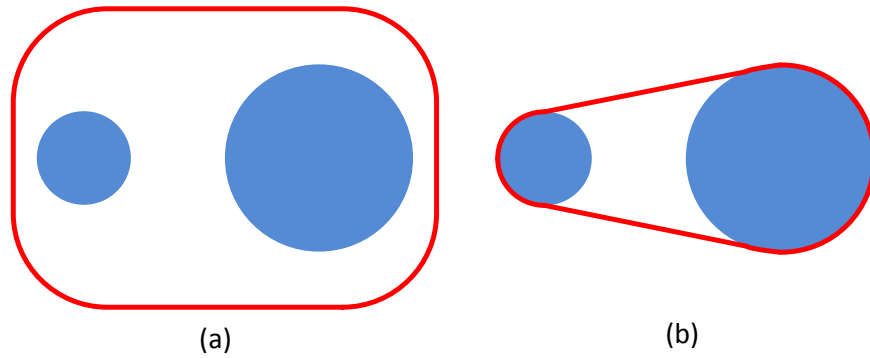
In Chapter 2, the method of segmentation using active contours for object detection and tracking was mentioned. This method uses an iteratively evolving contour in the image that separates different regions of interest.

Active contours can be expressed using one of the following two approaches [35]:

- Explicit or Lagrangian approach resulting in an interface known as *snake*.
- Implicit or Eulerian approach resulting in an interface called a *level set*.

Kass *et al* [40] initially introduced the concept of active snakes for expressing the contour in the image [36] in which a parameterised spline is guided in the image by a number of forces to a desirable position. These forces consist of internal forces that enforce intrinsic behaviour of the curve (such as smoothness), image forces that guide the curve towards desirable locations (such as edges) and external forces that may come from some user input or prior knowledge. These forces are imposed on the curve in the form of an energy function that measures the “fitness” of the snake and is minimised for desirable snake position and behaviour, for example, being on an edge in the image.

The major problem with active snakes is its inability to deal with changes in topology [37]. Figure 4-1 shows an example of this, where a snake that is designed to be attracted to edges in an image is used to segment two separate circles in an image. Assuming an initial position in Figure 4-1(a), the contour is unable to split to correctly segment both of the two circles, as shown in Figure 4-1(b).



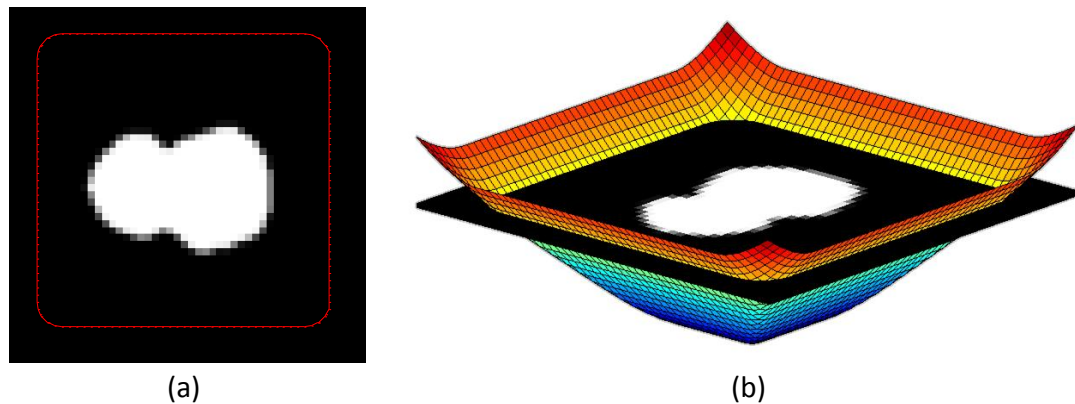
**Figure 4-1 - Inability of snakes to deal with changes in topology.** (a) is the initial contour in the image and (b) is the final contour after evolution

Level set methods were originally introduced by Osher and Sethian [38] as a means to evolve a contour with a speed proportional to its curvature. The main advantage of this method in an image segmentation context is that it allows for cusps, corners and automatic topological changes that are not allowed in active snakes [37].

Given an image  $I(x, y)$ , where  $(x, y)$  are image coordinates, a three-dimensional surface defined by a level set function  $\phi(x, y)$  is defined on top of it. The contour  $C$  is in the image and implicitly defined as the zeroth level set (hence the name) of the level set function  $\phi$ :

$$C = \{(x, y) | \phi(x, y) = 0\} \quad 4-1$$

Figure 4-2 assists visualisation of this expression where a contour in an image (Figure 4-2(a)) is defined as the part of the image that is cut by the level set surface at  $\phi = 0$ .



**Figure 4-2 - Example of a contour in an image (a) and the level set function that defines it (b)**

Figure 4-3 illustrates how this method of expressing the image contour deals with changes in topology. Simply by shifting the level set function up and down the resultant image contour is able to split and join thus allowing the method to deal with images such as that in Figure 4-1.

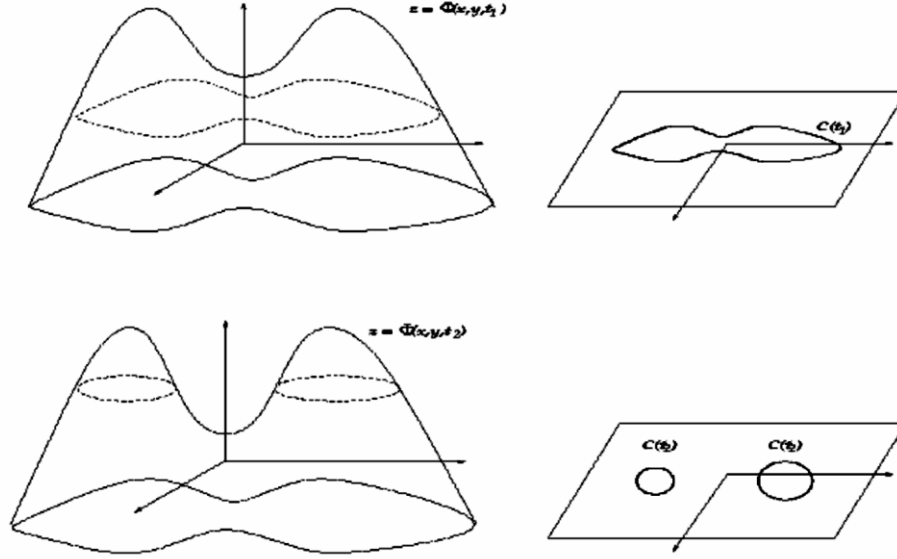


Figure 4-3 - Example of a 3D level set function intersecting with a 2D plane and its ability to handle changes in topology [39]

By inspection it is easy to see that an infinite number of functions would be able to produce the same contour in an image. To ensure a one-to-one mapping between the level set function and its corresponding contour, the level set function  $\phi$  is constrained to a *signed distance function*, that is:

$$|\nabla\phi| = 1 \text{ almost everywhere with } \phi > 0 \text{ inside the contour and } \phi < 0 \text{ outside the contour [40].}$$

At each pixel inside the contour, a signed distance function is equal to the distance from that pixel to the zero-level contour. At each pixel outside the contour it is equal to the negative of this distance. Some authors [10] choose to use the inverse of this and use  $\phi < 0$  inside the contour although this is simply a matter of preference. Unless otherwise stated, the convention of  $\phi > 0$  inside the contour is assumed.

## 4.2 Level Set Evolution

The level set function  $\phi$  is evolved over time in order to move its contour to regions of interest (i.e. around objects of interest). The level set function at any given time is expressed as  $\phi_t$  where  $t$



is an artificial time parameter. Evolution takes place by defining a differential equation that controls the evolution of  $\phi$ . This is of the form:

$$\frac{d\phi}{dt} = H \quad 4-2$$

Where  $H$  is dependent on the technique used. To evolve a given level set function using  $H$ , one can approximate  $\frac{\partial\phi}{\partial t}$  as:

$$\frac{\partial\phi}{\partial t}(x, y) = \frac{\phi_{new}(x, y) - \phi_{old}(x, y)}{\Delta t} \quad 4-3$$

Where  $\phi_{old}(x, y)$  is the current value of the level set function at  $(x, y)$ ,  $\phi_{new}(x, y)$  is the new value and  $\Delta t$  is a time-step parameter (usually less than 1 to ensure stability). Replacing 4-3 in 4-2 and rearranging, the new value of  $\phi$  can be calculated as:

$$\phi_{new}(x, y) = \phi_{old}(x, y) + H \Delta t \quad 4-4$$

Evolution may occur for a fixed number of iterations or when some conditions are met, for example,  $\phi$  no longer changes with each iteration.

#### 4.2.1 Reinitialisation

While a level set function may initially be defined as a signed distance function, evolution generally causes it to deviate from this. The function must then be reinitialised periodically throughout evolution by evolving it according to:

$$\frac{\partial\phi}{\partial t} = \text{sign}(\phi)(1 - |\phi|) \quad 4-5$$

## 4.3 Classical vs. Variational Level Set Methods

### 4.3.1 Classical Level Set Methods

The original implementation introduced in [38] attempts to simulate Active Snake behaviour using a level set function's contour. The contour is evolved according to active-snake-like forces  $F$  in the normal direction to the contour (in the image) to simulate the behaviour of active snakes. To do so the following differential equation is used:

$$\frac{\partial \phi}{\partial t} = F|\nabla \phi| \quad 4-6$$

This is known as the *classical formulation*. If  $F$  is positive at a particular point on the curve, the contour expands outward at this point in the image, conversely a negative  $F$  causes an inward expansion.

Inspecting this formulation, it is easy to see the reason behind ensuring  $\phi$  is a signed distance function, as  $|\nabla \phi|$  is always equal to 1, which simplifies calculations. Like active snakes, the force  $F$  is a function of both internal forces, which are dependent on the level set function alone and enforce certain characteristics on the contour; and external forces, which are dependent on the contour's relation to the underlying image and pull it towards certain features.

### 4.3.2 Variational Level Set Methods

Rather than predefine a set of forces in the image, variational level set methods seek to produce a level set function that *minimises* a predefined cost-function, more-specifically known as energy *functional*.

*A functional is defined as a mapping that takes the level set function as an input and returns a scalar value.*

This energy functional can be thought of as the measurement of the “fitness” of a particular level set function by penalising undesired behaviour. Generally the functional will incorporate certain criteria for the level set function that, when met, will produce a minimum in its output. By structuring the problem this way segmentation becomes an optimisation problem for which a level set function  $\phi$  with a minimum energy is the solution.

Variational methods deal with the behaviour of the entire level set function [35] rather than with points on the level set contour alone. This is especially necessary for methods that segment the image using properties of the *regions* contained by the contour rather than edges in the image.

While there are a number of methods of finding an optimal  $\phi$ , the most common is the Euler-Lagrange equation; a differential equation whose solutions are functions for which a given functional is stationary. Therefore by applying the Euler-Lagrange equation to a functional of  $\phi$ , one obtains a differential equation that evolves  $\phi$  in a direction that minimises this functional.

## 4.4 Common Level Set Formulations for Image Segmentation

While level set methods can be applied to any application that requires an evolving contour, the following formulations are commonly-known methods of applying them to the field of image segmentation.

### 4.4.1 Local/Edge-Based Segmentation

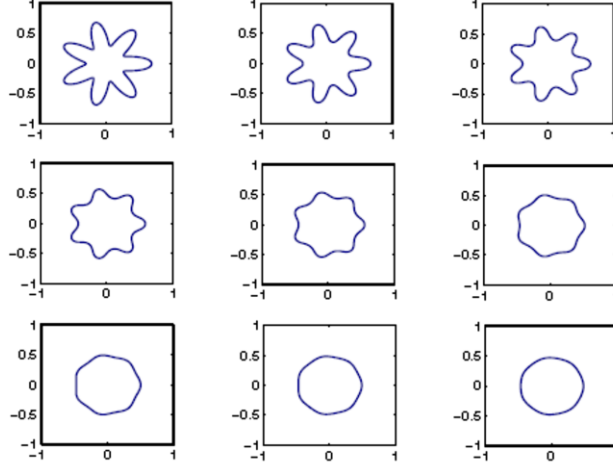
In the original level set implementation by Osher and Sethian [38] introduced in section 4.1 the authors sought to have a contour move with a velocity dependant on its mean curvature. This is a classical level set formulation with a force  $F$  expressed as:

$$F = \kappa = \operatorname{div} \left( \frac{\nabla \phi(x, y)}{|\nabla \phi(x, y)|} \right) \quad 4-7$$

The differential equation for this type of motion is:

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \operatorname{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) \quad 4-8$$

Figure 4-4 shows an example of a star-shaped contour that is undergoing purely mean curvature-based evolution. It has been shown that any shape undergoing this type of evolution collapses to a circle [41].



**Figure 4-4 - Star-shaped interface undergoing curvature-driven evolution.** *The tips of the star move inward, while the gaps in between them move outward [42]*

In order to apply this to a segmentation problem Caselles *et al* [43] introduce the *geometric active contour*. First, an edge detector function  $g$  is introduced:

$$g(I) = \frac{1}{1 + |\nabla \hat{I}|^p} \quad 4-9$$

Where  $\hat{I}$  is a smoothed version of the image  $I$  and  $p = 1$  or  $2$ . The term  $\nabla \hat{I}$  is the gradient of the image which is highest at edge pixels. While  $g$  tends to 1 when there are no edges present, it tends to zero when there are. The differential equation 4-8 is modified with this term and defined as:

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= g(I) |\nabla \phi| \operatorname{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + c g(I) |\nabla \phi| \\ &= g(I) (\kappa + c) |\nabla \phi| \end{aligned} \quad 4-10$$

Where  $c$  is a positive constant. This differential equation can be thought of as a combination of a constant outward force and mean curvature force that is modulated by the edge detector function so that the contour evolves to sit on strong edges. When the contour lies on strong edges,  $g$  tends to zero thereby keeping it stationary. While the constant force acts to inflate the contour within a region, it is clear from Figure 4-4 that the curvature-based force helps to keep it smooth.

#### 4.4.2 Regional Segmentation

While geometric active contours work well in images with strong edges, in many real-world applications this is not always the case. Chan and Vese [37] introduce a regional-based variational formulation that is designed to work with images without strong edges by *minimising the variation in pixel intensity inside and outside the contour*. By doing so, it is expected to move the level set contour around an image segment that is as homogenous in pixel intensity as possible.

The definitions of certain functions are important in defining regional variational functionals and thus are discussed first. The Heaviside function  $H$  is defined as:

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad 4-11$$

In a level set formulation, the Heaviside function is used to specify areas inside the contour, where  $H(\phi) = 1$ ; and outside the contour, where  $1 - H(\phi) = 1$ .

The Chan Vese energy consists of two internal energies  $E_{length}$  and  $E_{Area}$  that penalise the length of the contour and the area within it respectively (therefore favouring, small, short contours) and two external energies  $E_{Var\_inside}$  and  $E_{Var\_outside}$  that penalise variation in pixel intensity inside the contour and outside the contour respectively:

$$E_{CV} = \mu E_{length} + \nu E_{Area} + \lambda_1 E_{Var\_inside} + \lambda_2 E_{Var\_outside} \quad 4-12$$

Where  $\mu, \nu, \lambda_1$  and  $\lambda_2$  are parameters weighting importance of their respective penalty terms.

For an image domain  $\Omega$ , the inner and outer pixel-variation terms can be expressed as:

$$E_{Var\_inside} = \iint_{\Omega} |I - c_1|^2 H(\phi) dx dy \quad 4-13$$

$$E_{Var\_outside} = \iint_{\Omega} |I - c_2|^2 (1 - H(\phi)) dx dy$$

Note that the Heaviside function is used to control regions inside and outside the contour.  $c_1$  and  $c_2$  are the average pixel intensities inside and outside the level set contour respectively calculated as:

$$c_1(\phi) = \frac{\iint_{\Omega} IH(\phi)dxdy}{\iint_{\Omega} H(\phi)dxdy} \quad 4-14$$

$$c_2(\phi) = \frac{\iint_{\Omega} I(1 - H(\phi))dxdy}{\iint_{\Omega} (1 - H(\phi))dxdy} \quad 4-15$$

The resultant evolution equation that minimises the functional in equation 4-12 is as follows:

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left[ \mu \operatorname{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) - \nu - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2 \right] \quad 4-16$$

From comparison with the geometric active contour model it is clear that the first term is curvature dependant and seeks to enforce smoothness in the contour while the second is a constant contraction of the contour. Figure 4-5 shows segmentation of an image region that has weak edges using this equation. Eventually the variation of pixel intensities inside and outside the contour become equal and the contour remains stationary.

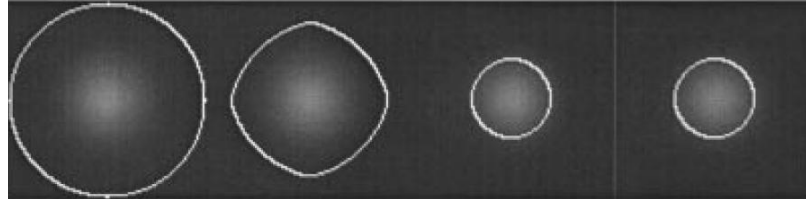


Figure 4-5 - Segmentation results using the Chan-Vese functional [37]

## 4.5 Shape Priors

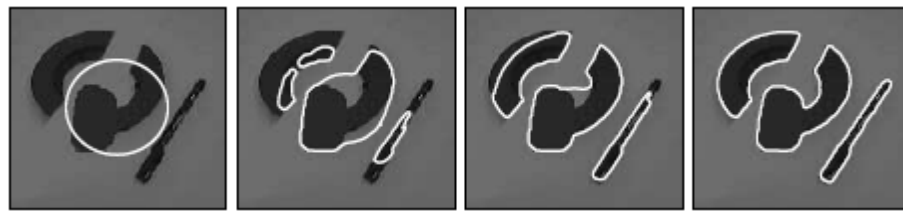
Active contours and specifically level set methods allow the use of other prior knowledge to aid segmentation results. Level sets methods with *shape priors* are geared toward segmenting objects of a predefined shape. It is fairly suitable to assume that in most applications, some knowledge of the shape of the objects of interest is available beforehand. The use of shape priors is especially useful when the segmentation target in question is corrupted, as is often the case in real-world applications (such as in maritime surveillance).

Figure 4-6 shows an original image (left) and a version that has been artificially corrupted by occlusion and deletion of certain parts of the object (right).

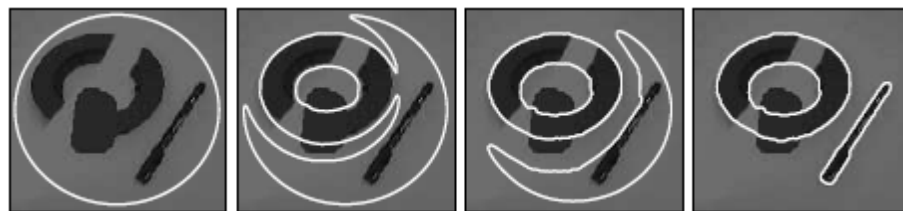


**Figure 4-6 - Original image (left) corrupted by artificial occlusion and deletion (right) [44]**

While normal segmentation of the corrupted image (Figure 4-7) incorrectly ignores the occluding pixels parts of the object, adding some prior knowledge of the rings shape to the segmentation (Figure 4-8) allows segmentation to obtain an output as if the corruption was not present.



**Figure 4-7 - Normal segmentation of corrupted image [44]**



**Figure 4-8 - Segmentation of corrupted image with prior knowledge of shape [44]**

Currently, there are two different ways to incorporate prior information about shape into segmentation. The first method modifies a variational energy functional designed for segmentation by adding an additional term that penalises deviation from a particular shape. The second method incorporates shape information into the functional directly by defining it as the output of some parametric function. Details of these methods are discussed next.

#### 4.5.1 Shape Energy Term

The majority of techniques that incorporate shape priors use a linear combination of a variational segmentation functional (as discussed above in section 4.4.2) and a shape difference term [45]. The purpose of the shape difference term is to penalise level set contours that deviate from a predefined shape. This is similar to the various other penalisation terms already seen in section 4.4.2.

A rudimentary example of a shape difference term introduced by Paragios and Rousson [46] is the squared difference between the segmenting level set function  $\phi$  and a pre-defined level set function that incorporates the desired shape  $\psi$ :

$$E_{shape}(\phi, \psi) = \iint_{\Omega} (\phi(x) - \psi(x))^2 dx dy \quad 4-17$$

This term is added to the segmentation-based functional (such as the Chan-Vese functional  $E_{CV}$ ). It is often multiplied by a weighting factor  $\alpha$  to control the balance between the two terms:

$$E(\phi, \psi, I) = E_{CV}(\phi, I) + \alpha E_{shape}(\phi) \quad 4-18$$

Chan and Zhu [45] define the shape distance as:

$$E_{shape}(\phi, \psi) = \iint_{\Omega} (H(\phi) - H(\psi))^2 dx dy \quad 4-19$$

Where  $H$  is the Heaviside function. This term is symmetric to  $\phi$  and  $\psi$  and further does not require that either function is a signed distance function. More importantly it is not dependent on the size of the domain  $\Omega$ .

The problem with this type of method is that it requires that the level set function that encodes the prior shape lie directly over the object to be segmented in the image. To solve this, Chan and Zhu introduce a shape equivalence, which states that two objects' shape are considered the same regardless of their scale, rotation or translation with respect to one another.



More formally, for two equivalent shapes encoded by signed distance functions  $\phi_1$  and  $\phi_2$  there exists a four-tuple  $(a, b, r, \theta)$  such that:

$$\phi_2(x, y) = r\phi_1\left[\frac{(x-a)\cos\theta + (y-b)\sin\theta}{r}, \frac{-(x-a)\sin\theta + (y-b)\cos\theta}{r}\right] \quad 4-20$$

Where  $a$  and  $b$  are horizontal and vertical translation parameters,  $r$  a scaling factor and  $\theta$  an angle of rotation. Therefore given a signed distance function in a particular shape, all shape-equivalent functions can be generated by choosing  $(a, b, r, \theta)$ .

This equivalence can be used as a transformation, where  $\phi_2$  is a scaled, translated and rotated version of  $\phi_1$ . Chan and Zhu introduce the original shape prior as  $\psi_0$ , where  $\psi$  is now a transformed version of  $\psi_0$  according to equation 4-20. The shape energy term then becomes:

$$E_{shape}(\phi, \psi_0, a, b, r, \theta) = \iint_{\Omega} [H(\phi) - H(\psi_{\psi_0, a, b, r, \theta})]^2 dx dy \quad 4-21$$

This energy term combined with the original segmentation energy functional is minimised with respect to its input parameters. As  $\psi_0$  is constant,  $\phi$ ,  $a$ ,  $b$ ,  $r$  and  $\theta$  are all evolved according to their own update equations. As  $\phi$  evolves, its contour will move in a direction that minimizes the image energy while still being held in shape by  $\psi$ . As the pose parameters evolve, they will ensure that  $\psi$ , while still remaining in the original shape of  $\psi_0$ , keeps up with  $\phi$  in its efforts to minimize the segmentation term.

#### 4.5.2 Parametric Functional

Parametric methods of incorporating shape information impose this knowledge directly into the level set by defining the level set as the output of some parametric function. This parametric function is defined by a set of parameters and segmentation energy functionals are then minimised with respect to these parameters rather than the level set function itself.

Consider a toy example where it is known that the shape to be segmented is a circle. Here the level set can be defined as a parametric function:

$$\phi(x, y)_{a, b, r} = (x - a)^2 + (y - b)^2 - r^2 \quad 4-22$$

Where the circle's centre coordinates  $(a, b)$  and its radius  $r$  are the only parameters required to define any circle at any location in the image. Rather than optimising a segmentation functional with respect to the *entire* function  $\phi$ , it is optimised with respect the parameters  $a, b$  and  $r$ .

Tsai *et al* [10] introduce a parametric level set function for the use of shape priors that is an affine transformed version of a prior-known shape, defined in a level set function  $\psi$ .

The segmenting level set function  $\phi$  is thus controlled by the pose parameters  $\mathbf{p} = [a, b, h, \theta]$  where  $a$  and  $b$  control the horizontal and vertical translation of the shape prior,  $h$  controls scaling of the shape prior and  $\theta$  its rotation in the image. This mapping of the shape prior function  $\psi$  to the segmenting function  $\phi$  is shown in Figure 4-9.



**Figure 4-9 - Example of a shape prior encoded in function  $\psi$  manipulated using  $\mathbf{p}$  to produce a level set function  $\phi$  with associated contour in the image**

Parametric models allows for a faster evolution that is less prone to getting stuck in local minima as the energy is minimised directly by manipulating a few parameters rather than the entire contour.

The most pertinent benefit of using this type of implementation is that it does not require function re-initialisation as discussed in section 4.2.1. The resultant segmenting level set function is always a transformed version of an original signed distance function, which itself is thus a signed distance function.

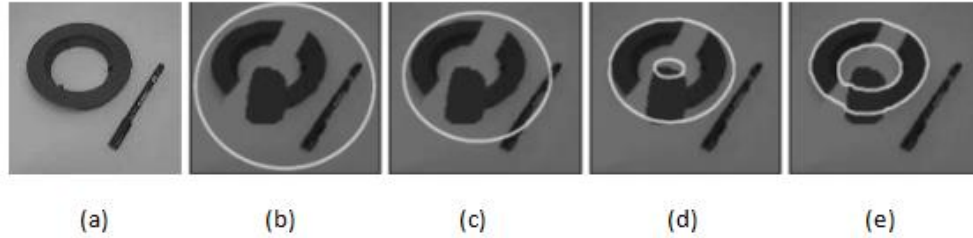
Additionally the limited degrees of freedom allows for more “brute force” numerical methods of energy optimization, which allow the contour to evolve according to any arbitrary energy functional without the need for explicit differentiation. Details of this will be referred to later in the dissertation.

### 4.5.3 Multiple Shape Priors

The methods discussed so far assume that a single shape prior is used unconditionally in segmentation. While this may not be a problem if one is to segment a single known object, it is impossible to segment objects that are unknown or to segment other known objects. In a tracking example, one might want to incorporate many different views of a three dimensional object that one is searching for, or a set of different shapes to account for the variability in targets that one might want to track.

#### 4.5.3.1 Selective and Competing Shape Priors

Figure 4-10 shows segmentation using a single shape prior. The original image is shown in (a) and the shape of the ring is used as a prior. Subfigures (b) to (e) show successful segmentation of the corrupted image despite occlusion. Although it is a valid target, the pen is ignored as it is not part of the ring-shaped prior. In order to successfully segment both objects it is desirable to have a *selective* shape prior term which turns shape prior segmentation on or off depending on the object to be segmented.



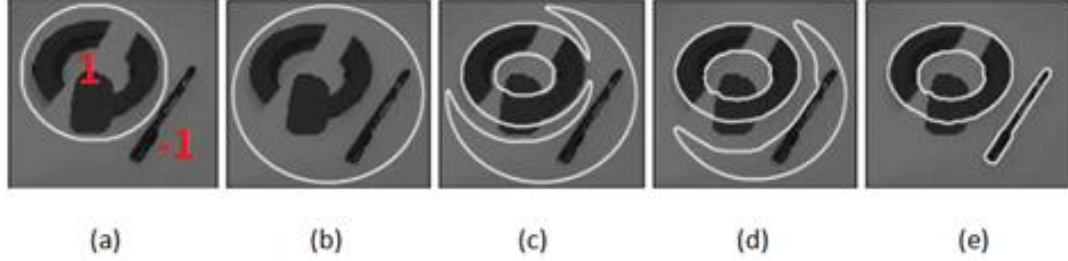
**Figure 4-10 - Segmentation with a shape prior.** Sub-figure (a) is the original image without removal or occlusion [44]

Cremers *et al* [44] approach this problem by modifying a shape difference term with a labelling function  $L(x, y)$  defined over the entire image that controls whether shape knowledge is used in certain sections of the image or not:

$$E_{shape}(\phi) = \iint_{\Omega} (\phi(x) - \psi(x))^2 (L + 1)^2 dx dy \quad 4-23$$

If  $L = 1$  in a certain region, the coefficient  $(L + 1)^2$  is equal to 4 and the shape prior segmentation occurs as usual. If  $L = -1$ , this coefficient is equal to 0 and the shape term becomes inconsequential.

Figure 4-11 shows segmentation using this new functional term. The value of  $L$  is initially predefined and is shown in (a) to be 1 within the white circle and -1 elsewhere. Now the level set function successfully segments the ring with the addition of the shape prior term and the pen using pure segmentation without prior shape knowledge.



**Figure 4-11 - Segmentation with shape prior and labelling function.** *The labelling function has values shown in sub-figure (a) [44]*

Obviously the need to manually specify the labelling function before segmentation is undesirable. To overcome this limitation, the authors introduce a dynamic labelling function that is evolved concurrently with the level set function. The shape prior term becomes:

$$E_{shape}(\phi, L) = \iint (\phi(x) - \psi(x))^2 (L + 1)^2 dx dy + \iint \lambda^2 (L - 1)^2 dx dy + \gamma \iint |\nabla H(L)| dx dy \quad 4-24$$

This is minimised with respect to both the level set function and the labelling function simultaneously. The first two terms enforce the shape prior in regions where the level set function is similar to the prior, or more specifically:

$$L = \begin{cases} 1 & \text{if } |\phi - \psi| < \lambda \\ -1 & \text{if } |\phi - \psi| > \lambda \end{cases} \quad 4-25$$

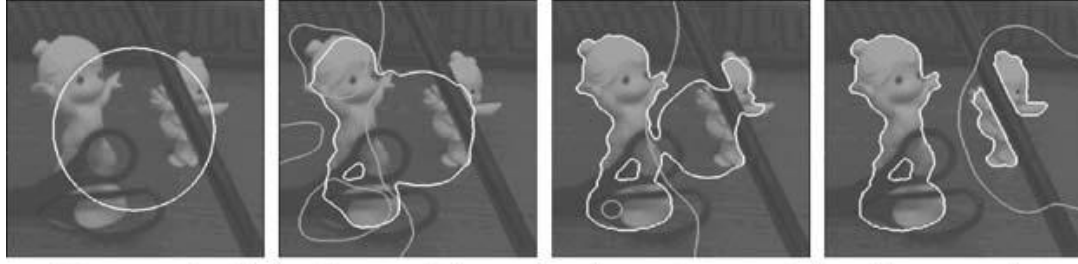
Where  $\lambda$  is a predefined threshold and  $\gamma$  is a weighting term. The last term enforces topological compactness of regions.

In order to use multiple priors, Cremers *et al* [47] build on their previous work in [44]. The energy functional in equation 4-24 can initially be extended to segment two known objects, associated with level set functions  $\psi_1$  and  $\psi_2$ :

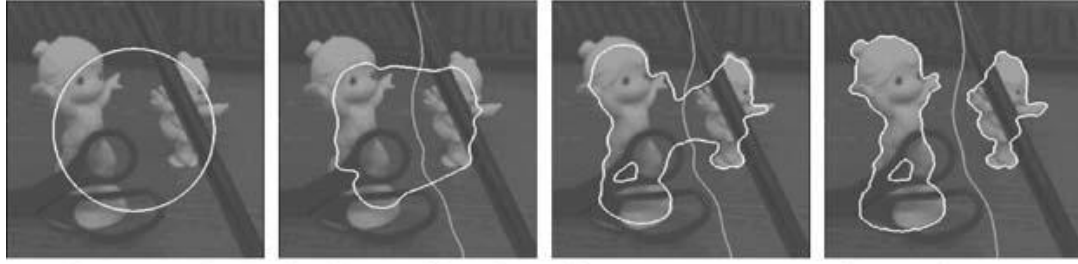
$$E_{shape}(\phi, L) = \frac{1}{\sigma_1^2} \iint (\phi - \psi_1)^2 (L + 1)^2 dx dy + \frac{1}{\sigma_2^2} \iint (\phi - \psi_2)^2 (L - 1)^2 dx dy + \gamma \iint |\nabla L| dx dy \quad 4-26$$

Where  $\sigma_i^2 = \int \psi_i^2 dx - (\int \psi_i dx)^2$  is the variance of the respective function. The labelling function  $L$  is now driven by two *competing* shape priors; each image location will either be assigned to use one shape or the other. In [44] the labelling function  $L$  had two values: 1 in regions where the shape prior was to be enforced (the known object), and  $-1$  in regions where it was not (the background or unknown objects). In this case the labelling function is  $-1$  for the first object and 1 for the second.

A comparison of the segmentation with a single prior and the background (no shape prior) and with two competing shape priors is shown in Figure 4-12. For a single shape prior, the left statue is successfully segmented regardless of occlusions while the right statue is treated as an unknown object and is unsuccessful. With two competing shape priors, both statues are successfully segmented regardless of occlusions.



Dynamic Labeling with a single prior and background.



Dynamic Labeling allowing for two competing priors.

Figure 4-12 - Segmentation comparison of image with single shape prior vs. two competing priors [47]

For the general case, where a number of competing shapes are concerned, the labelling function must be modified to be able to work with multiple objects. The authors introduce a vector-valued labelling function:

$$\mathbf{L}(x) = (L_1(x), \dots, L_n(x)) \quad 4-27$$

A polytope (an  $n$ -dimensional polygon) with  $m = 2^n$  vertices of at  $[-1, +1]^n$  are used to encode  $m$  different regions.  $\chi_i, i = 1, \dots, m$  are indicator functions for each region, where each indicator function is at its minimum when in a region that  $L$  encodes its specific shape. For four different shapes,  $\mathbf{L}$  would have four different values:  $(-1, -1), (-1, 1), (1, -1)$  and  $(1, 1)$ .

The four indicator functions would be:

$$\begin{aligned} x_1(\mathbf{L}) &= \frac{1}{16} (L_1 - 1)^2 (L_2 - 1)^2 \\ x_2(\mathbf{L}) &= \frac{1}{16} (L_1 + 1)^2 (L_2 - 1)^2 \\ x_3(\mathbf{L}) &= \frac{1}{16} (L_1 - 1)^2 (L_2 + 1)^2 \\ x_4(\mathbf{L}) &= \frac{1}{16} (L_1 + 1)^2 (L_2 + 1)^2 \end{aligned} \quad 4-28$$

Or generally:

$$x_i(\mathbf{L}) = \frac{1}{4^n} \prod_{j=1}^n (L_j + l_j)^2 \quad 4-29$$

With  $l_j \in \{-1, 1\}$ . The new shape energy becomes:

$$\begin{aligned} E_{shape}(\phi, \mathbf{L}) = & \sum_{i=1}^{m-1} \iint \frac{(\phi - \psi_i)^2}{\sigma_i^2} x_i(\mathbf{L}) dx dy + \iint \lambda^2 x_m(\mathbf{L}) dx dy \\ & + \sum_{i=1}^m \iint |\nabla L_i| dx dy \end{aligned} \quad 4-30$$

Where each  $\psi_i$  corresponds to a known shape with variance  $\sigma_i$ .

Figure 4-13 shows an example of segmentation of a corrupted image using multiple shape priors. The top row of images shows the evolution of the segmenting contour. Even though the bottom three statues are corrupted in the image, they are successfully segmented due to the enforcing of a shape prior with each. The top statue does not have a prior and is treated as an unfamiliar object. This result can be compared to segmentation without a prior in the last image of the bottom row, which incorrectly (blindly) segments each shape. Without the shape prior, the occluded parts of the objects were incorrectly excluded from the segmentation.

The labelling is shown on the bottom row of Figure 4-13. The third image of the row shows how each statue occupies a region that is characterised by a different value of the labelling function.

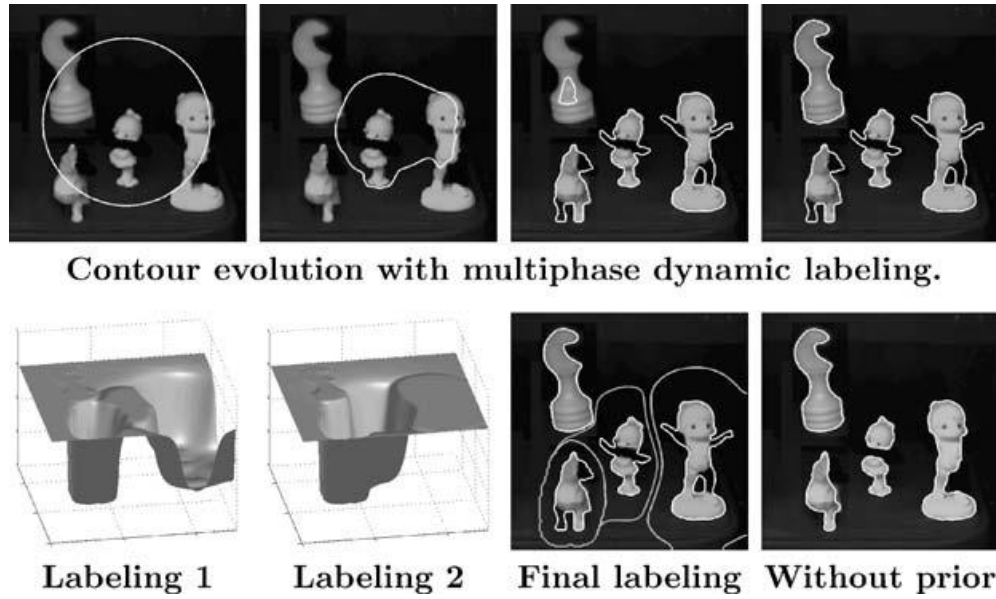


Figure 4-13 - Segmentation of corrupted image using multiple shape priors [45]

#### 4.5.3.2 Shape Model

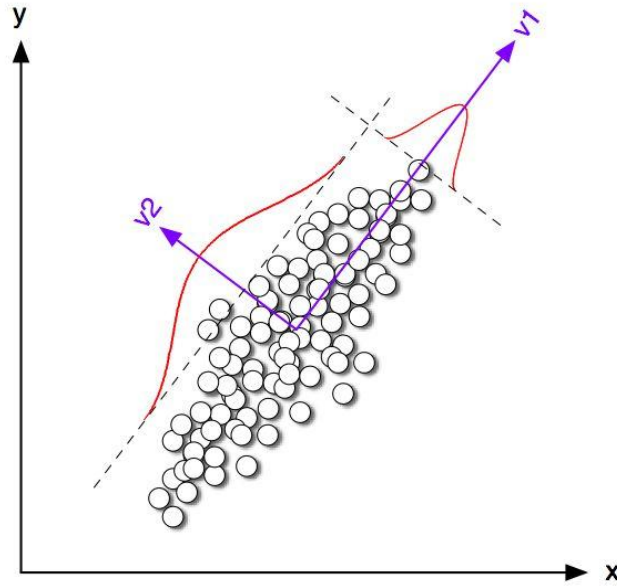
The following methods incorporate the use of multiple shape priors by defining a *shape model* rather than using a single prior known shape. In this section, the shape model may be created using principle component analysis, Gaussian modelling or kernel density estimation. The level set segmentation uses this model which reflects the shapes of an *entire training set*.

##### i. Principle Component Analysis

Tsai *et al* [10] propose parametric level set function that is a combination of principle components from a set of multiple training shapes. Principle component analysis (PCA) [48] is a method of separating observations of possible correlated variables into values of *principal components* which are uncorrelated. The first principle component accounts for the most variation in the data with each subsequent component accounting for the greatest possible amount of remaining variation.

Figure 4-14 shows an example of two dimensional data that has been separated into two principle components  $v_1$  and  $v_2$ . It is easy to see that the majority of variation occurs along the first principle component  $v_1$ .





**Figure 4-14 - Example of Principle Component Analysis in two dimensions [49].** *The variance of the data in the original Cartesian space  $(x, y)$  is best captured by the basis vectors  $v_1$  and  $v_2$  in a rotated space*

The number of principal components used to express the observations is equal to or less than the original number of variables. PCA is commonly used as a method of dimensionality reduction when the observations are expressed with fewer principal components than original variables. In Figure 4-14 for example, if one were to express the given observations using only  $v_1$  as a basis they would not deviate very much from their original locations.

To find principle components of a particular data set eigenvectors and eigenvalues of the data are extracted from its covariance matrix via Singular Value Decomposition (SVD) [49].

In order to model a set of training shapes using this method, the shapes are aligned with one another such as those in Figure 4-15. These are embedded in signed distance functions that are sampled on a grid at fixed locations.



**Figure 4-15 - Example set of binary images after alignment [10]**

Notice in Figure 4-15 that training shapes exhibit similar behaviour with small deviations between them. This “principle-shape” is modelled by applying PCA to the set.

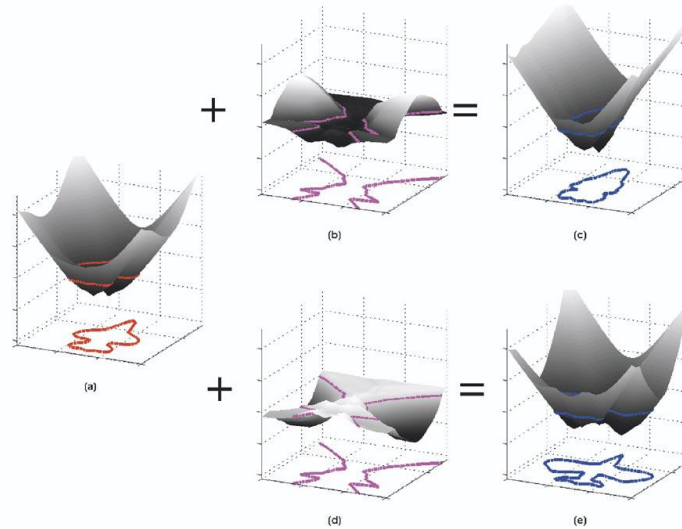
Each shape function is sampled and reduced to a single-dimensional vector representing an observation like those in Figure 4-14. PCA is then applied to the set of observations, the output of which is a “mean-shape”  $\bar{\Phi}$  and a set of “principal/Eigen-shapes”  $\{\Phi_1, \Phi_2, \dots, \Phi_n\}$ .

These Eigen-shapes are weighted and added to the mean shape to produce a parametric level set function that is capable of transforming itself into one or a combination of training shapes depending on the weighting parameters:

$$\phi[\mathbf{w}] = \bar{\Phi} + \sum_{i=1}^k \omega_i \Phi_i \quad 4-31$$

Where  $k \leq n$  can be determined empirically, depending on the accuracy required.  $\mathbf{w} = [\omega_1, \omega_2, \dots, \omega_n]$  are the weights for each Eigen-shape and the variance for each weight  $\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$  is equal to its corresponding eigenvalue earlier.

Figure 4-16 shows two examples from [10] of how adding or subtracting  $\sigma_1 \Phi_1$  to the mean level set function can result in approximates of two of the training examples shown in Figure 4-15.



**Figure 4-16 - Three dimensional example of shape model for the fighter jet training set.** (a) is the mean level set function, (b) is  $+\sigma_1 \Phi_1$ , (d) is  $-\sigma_1 \Phi_1$  and (c) and (e) are the results of their summations with the mean level set function [10]

Similarly to the methods described in section 4.5.2, an energy functional (such Chan Vese) can be minimised with respect to the parameters  $\mathbf{w} = [\omega_1, \omega_2, \dots, \omega_n]$ .

## ii. Gaussian Modelling

Rousson *et al* [49, 53] create a model of a set of prior shapes by assuming that shape priors follow pixel-wise Gaussian distributions. The meaning of this is as follows:

- Given a sample set of aligned signed distance functions  $\{\hat{\psi}_1, \hat{\psi}_2, \dots, \hat{\psi}_n\}$ .
- The values the functions  $\{\hat{\psi}_1, \hat{\psi}_2, \dots, \hat{\psi}_n\}$  at each pixel  $(x, y)$  form a Gaussian distribution.

A shape model may then be created by creating a Gaussian model at each of the pixel locations, similarly to single Gaussian background model in section 3.3.

The probability  $p_M$  of a particular  $\phi$  value at  $(x, y)$  can then be calculated as:

$$p_M(\phi(x, y)) = \frac{1}{\sqrt{2\pi}\sigma_M(x, y)} e^{\frac{-(\phi(x, y) - \mu_M(x, y))^2}{2\sigma_M^2(x, y)}} \quad 4-32$$

Where  $\mu_M$  and  $\sigma_M^2$  are the Gaussian model mean and variance values at pixel  $(x, y)$ . In order to create an energy term, unlikely level set functions must be penalised. This is achieved by using the negative logarithm of this function, which is greater for unlikely level set functions. The shape energy term becomes:

$$E_{shape}(\phi) = - \iint_{\Omega} \log[p_M(\phi(x, y))] dx dy \quad 4-33$$

Which may be added to a segmentation functional (such as the Chan-Vese functional) as shown in equation 4-18.

## iii. Kernel Density Estimation

Cremers *et al* [50] identify two major disadvantages in using PCA or Gaussian modelling to encode the training set:

- Previous methods assume the training shapes (for example those in Figure 4-15) are fairly similar and form a pixel-wise Gaussian distribution. This is not always the case in practical applications, especially when encoding multiple views of a complex object.
- They assume the shapes are represented by signed distance functions; the combination thereof (as seen in equation 4-31) is not linear. Therefore the mean or a linear combination of signed distance functions isn't necessarily a signed distance function.

Cremers *et al* decide to apply the use of a kernel density estimator (described earlier in Chapter 3) to produce a statistical shape model that can model fairly distinct training shapes and isn't required to be linear.

A shape difference measure that is based on Chan and Zhu's work [45] is defined as:

$$d^2(\phi_1, \phi_2) = \iint_{\Omega} (H(\phi_1) - H(\phi_2)) dx dy \quad 4-34$$

For a set of  $N$  training shapes defined by signed distance functions  $\{\psi_1, \psi_2, \dots, \psi_N\}$ , the probability  $P$  of a particular segmenting level set function  $\phi$  can be estimated as:

$$P(\phi) \propto \frac{1}{N} \sum_{i=1}^N \exp\left(-\frac{1}{2\sigma^2} d^2(H(\phi), H(\psi_i))\right) \quad 4-35$$

This probability increases the closer  $\phi$  is to any of the pre-defined training shapes.  $\sigma$  is the kernel width. Choosing the kernel width is non-trivial and the authors choose to fix it to be the mean squared nearest neighbour distance:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N \min_{j \neq i} d^2(H(\phi), H(\phi_j)) \quad 4-36$$

This ensures that on average the next training shape is within one standard deviation of each Gaussian. In similar fashion to the Gaussian model described in the previous sub-section, this probability is maximised by minimising its negative logarithm. So the shape energy becomes:

$$E_{shape}(\phi) = -\log(P(\phi)) \quad 4-37$$

Which may be added to a segmentation functional (such as the Chan-Vese functional) as shown in equation 4-18. An example training set used for segmentation is shown below in Figure 4-17. Notice that training shapes can be quite different from one another compared with those in the above methods.



Figure 4-17 - Various samples from the training set used in Figure 4-19 [50]

A comparison of segmentation of a partially occluded walking person using purely intensity-based methods is shown in Figure 4-18 while segmentation with the addition of shape-based energies using the above training set is shown in Figure 4-19. Notice that the purely intensity-based functional fails to differentiate between the occlusion and the person and segments darker patches cast by the person's shadow whereas the prior based functional does not.

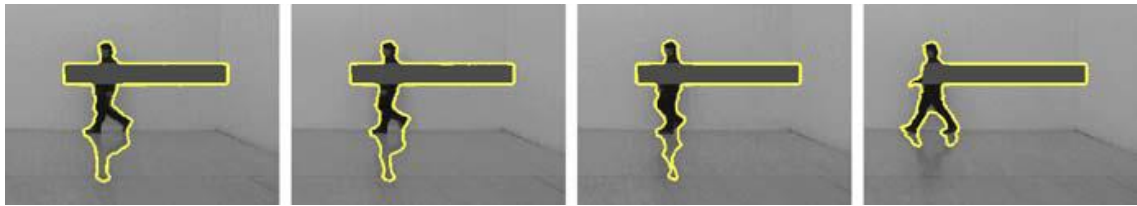


Figure 4-18 - Final segmentation results for various frames of an occluded walking person segmented with a purely intensity functional [50]



Figure 4-19 - Final segmentation results for various frames of an occluded walking person segmented with using a functional that includes the shape model [50]

## 4.6 Conclusion

This chapter introduces the concept of level set methods for segmentation. Theory behind the method is presented, including common variants of the method used for segmentation using edges in an image or regional properties within the segmenting contour. The use of prior knowledge of an object's shape to aid segmentation is then explored. Various techniques of incorporating this knowledge into the method are discussed, followed by a number of additions to allow segmentation to move from using a single shape to an entire set of training shapes, or selectively use the knowledge depending on the context in which segmentation is used.

## **Chapter 5 - Proposed Model**

### **5.1 Introduction**

The object detection and object tracking methods used in Szpak and Tapamo's work [8] have been introduced and extensively discussed in Chapter 3 and Chapter 4. The fundamental contribution of the work presented in this dissertation is a method of incorporating shape knowledge into a system similar to [8] and so a number of methods for doing this have also been presented in Chapter 4.

This chapter covers explicit details of the system that will use this shape knowledge in a maritime environment. It covers an overview of the system model used with main functional units and how they are connected, the level set algorithm used for object detection and tracking and finally specifics of how these units operate individually to produce the final video tracking system. The chapter also provides pseudo-code which clarifies algorithms used. This chapter does not deal with the actual final system model, as it presents a number of choices for each subsystem used and the pertinent parameters that define their behaviour. These subsystems and their parameters are rather chosen optimally in Chapter 6.

### **5.2 Model Overview**

There are two main types of video tracker architecture. These include architectures that apply detection and tracking separately, and those that perform them jointly [12].

In the first case, possible object regions are produced by the object detector, and the object tracker makes correspondences between these regions from frame to frame. This concept is illustrated in an example in Figure 5-1 where an object detector finds objects in two separate frames, while the object tracker is required to separately decide which of the detected objects match up from frame to frame.

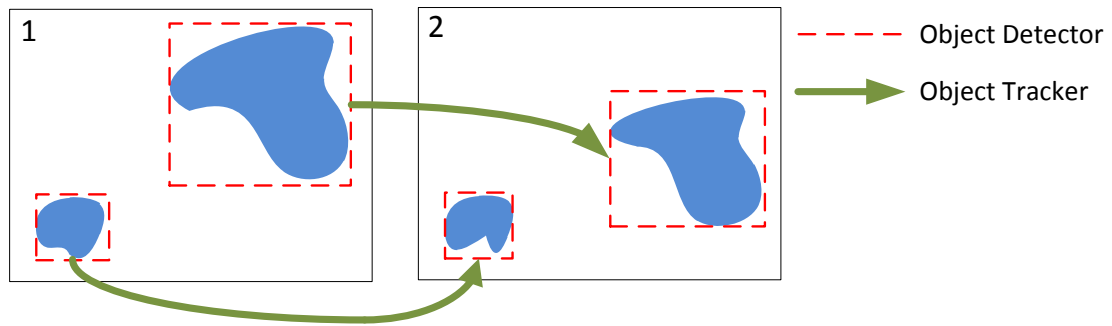


Figure 5-1 - Separate object detector and tracker architecture

In the second case the object regions and their correspondences are jointly estimated by keeping object and region information from previous frames, and simply updating them for the current one. This is illustrated in an example in Figure 5-2 where the detected boundary for a particular object in the first frame is kept in the second. This boundary is then updated to its new location in the image.

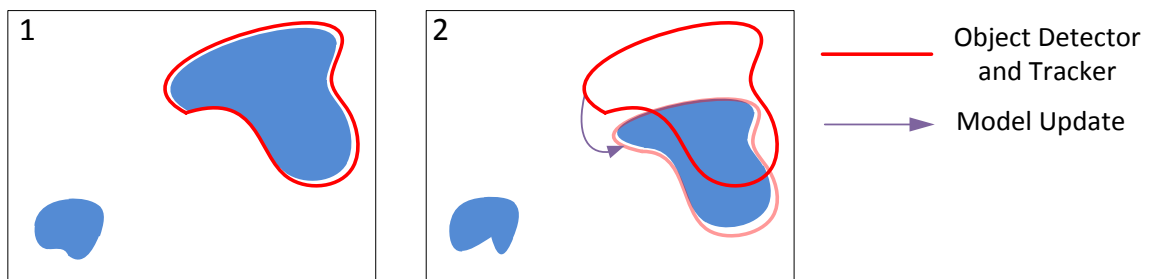


Figure 5-2 - Joint object detector and tracker architecture

Level set tracking falls into the latter architecture of the above examples. Here the level set segmentation is run on each frame where the starting contour for each particular frame is the final contour of its predecessor [55, 56].

As the proposed system uses a level set method for tracking it is based largely around the second architecture. There is, however, a single difference in that the level set contour used is unable to detect objects by itself and must rely on a separate sub-system to initialise it. While this sub-system mainly serves as a *tracker initialisation stage*, it is theoretically a form of *object detection* and will thus be labelled as such. It should be emphasised that unlike the object detection stage of the separate detector/tracker architecture, this sub-system does not operate on every frame.

An example of the system functionality is shown in Figure 5-3. Here the object detector is required to place the boundary around an object of interest in the first frame. Thereafter, tracking continues as it would in a joint tracker and detector architecture.

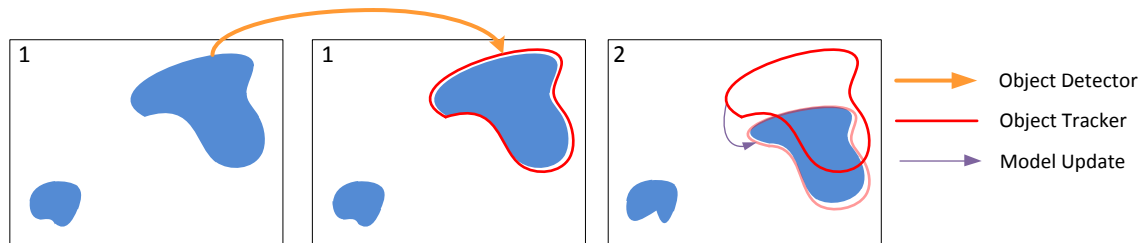


Figure 5-3 - Example of system functionality

While level set segmentation forms an integral part of the object tracking system, it also forms a part of the object detector, the details of which are discussed later. An overview of the entire proposed system is shown in Figure 5-4.

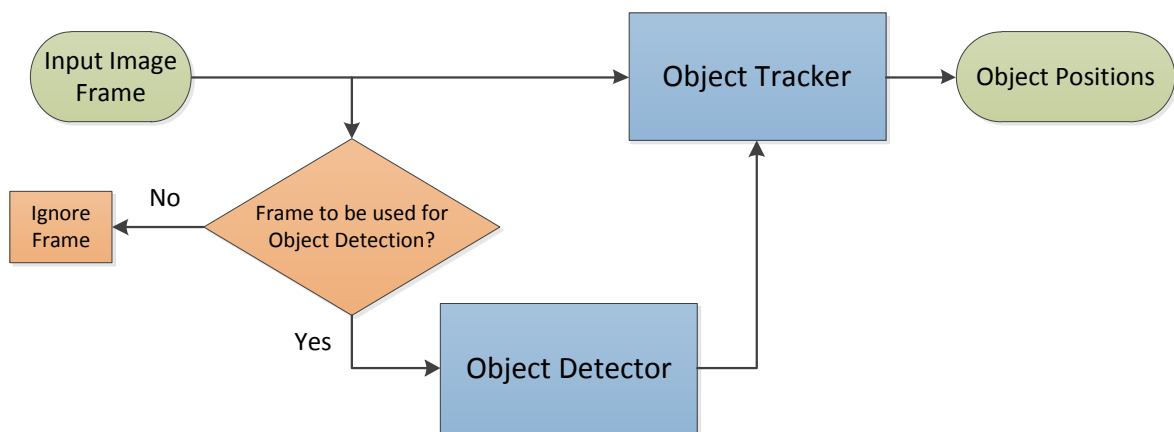


Figure 5-4 - Proposed system



The input to the system is a sequence of grey scale image frames from a video of a maritime scene. The output of the system is ideally the same set of image frames with various maritime objects of interest highlighted by a level set contour as illustrated in red by Figure 5-5.



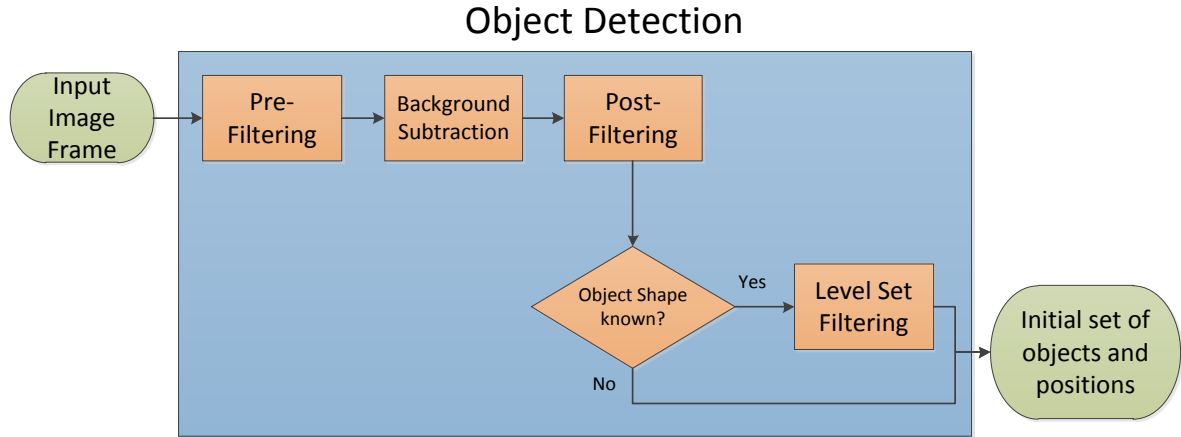
Figure 5-5 - Example output of the system

An overview of how the system functions is as follows:

- The object tracker does not run until it has received a set of initial object positions from the object detector.
- The object detector uses a background subtraction algorithm that only uses input frames periodically at a fixed spacing. Only once it has filled a buffer of frames is it able to produce an output and so until then the tracker, and thus the system, has no output.
- Once the buffer is full, the detector is able to output a set of objects to the tracker.
- Once it has obtained a set of initial object positions, the object tracker continues to track these objects.

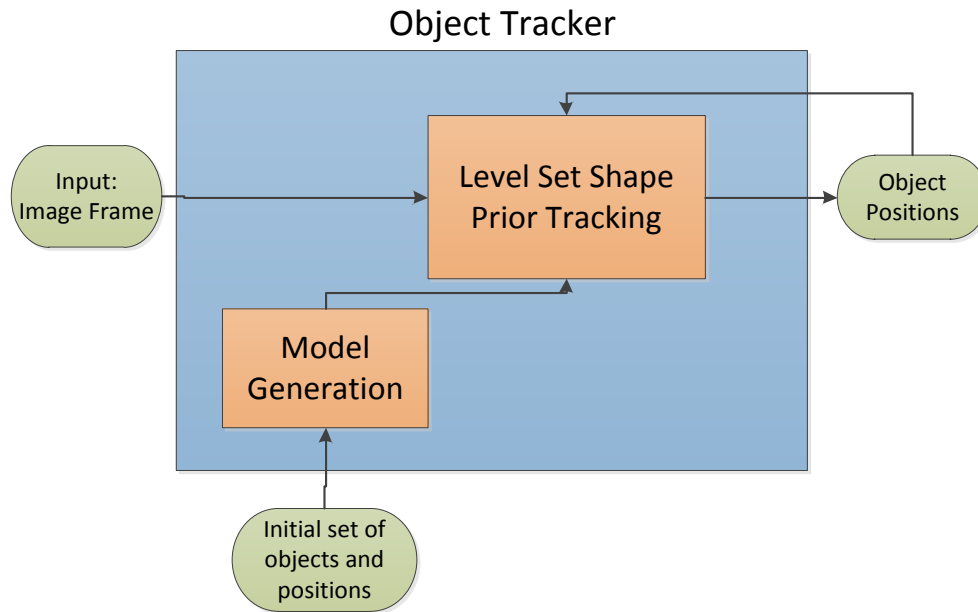
The object detector (Figure 5-6) consists of a *pre-filtering* stage, followed by a *background subtraction* algorithm. The resultant binary images are then filtered again (called *post-filtering* in this dissertation) to remove unwanted binary pixels in the image.

It is possible to use the binary output of this stage directly as the input to the object tracker. However, if it is desired by the user to find a particular shape that is known beforehand, binary level set shape prior segmentation can further be applied (called *level set filtering* in this dissertation) before the input to the object tracker. This attempts to find the shape in question in the noisy binary image and outputs its estimated position and pose in the frame.



**Figure 5-6 – Details of object detector**

The object tracker (Figure 5-7) extracts features of the detected objects in the first frame after detection to create a model for each object. For each subsequent frame the level set tracking algorithm uses this model, combined with its shape, to track the object. This information is fed back into the tracker for use in tracking the object in the next frame.



**Figure 5-7 - Details of object tracker**

The rest of the chapter is structured as follows: The level set shape prior framework that forms the basis of the system is first covered in detail. This is followed by the modifications that allow the use of the custom energy functionals required to be used with the framework. The manner with

which the modified level set segmentation algorithm is used in the object detector and tracker stages is then covered.

### 5.3 Level Set Segmentation Algorithm

While Szpak and Tapamo [8] used a general level set segmentation algorithm based around the work of Chan and Vese [37], the work presented here deals with incorporating shape knowledge into the system and so a different method is selected.

#### 5.3.1 Tsai et al's Level Set Algorithm

The work by Tsai *et al* [10] was chosen as the basis for the level set segmentation algorithms in the system. Tsai *et al*'s work uses a parametric functional, the benefits of which are discussed in section 4.5.2.

As discussed in section 4.5.3.2, part of Tsai *et al*'s work [10] deals with the incorporation of multiple shape priors in segmentation. For simplicity, however, the system described in this dissertation has been designed to use a single shape prior. For each sequence that is tested, the desired shape prior is manually set to the shape of an object that appears in that sequence.

The original work manipulates the level set function  $\phi$  using two sets of parameters: Parameters controlling its shape (see section 4.5.3) and parameters controlling its pose in the image. Simplification to a single shape prior allows the level set function to only be manipulated using the latter set of *pose* parameters (i.e. the one shape is known in advance and is given by the shape prior).

The set of pose parameters  $\mathbf{p} = [a \ b \ h \ \theta]^T$  is first introduced, where  $a$  and  $b$  control horizontal and vertical translation respectively,  $h$  the scaling and  $\theta$  the rotation. The level set function can be parameterised in terms of these pose parameters and this is now explained.

Consider a level set function  $\psi$  that has a contour in the form of the desired shape prior, the level set function  $\phi$  can be defined as a translated, scaled and rotated version of this original function:

$$\phi[\mathbf{p}](x, y) = \psi(\tilde{x}, \tilde{y}) \tag{5-1}$$

Where  $\tilde{x}$  and  $\tilde{y}$  are calculated according to:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = T[\mathbf{p}] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}}_{M(a,b)} \underbrace{\begin{bmatrix} h & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{H(h)} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta)} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad 5-2$$

As a single shape prior is used, the surface  $\psi$  remains static throughout segmentation. This means that the final level set function  $\phi$  is merely a translated, scaled and rotated version of some static shape and no longer able to move freely in the image as it does in other methods (see Chapter 4). An illustration of this definition is shown in Figure 5-8, where the contour in the image is dependent on  $\psi$  (the shape prior) and the pose parameters  $\mathbf{p}$ .

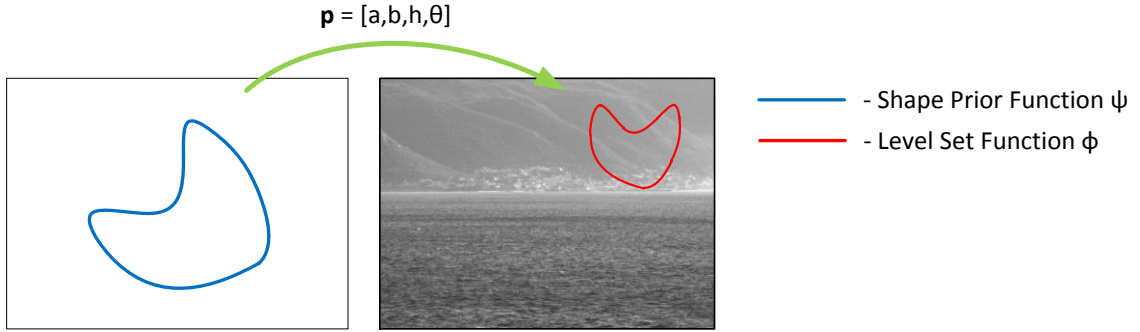


Figure 5-8 - Example of a shape prior encoded in function  $\psi$  manipulated using  $\mathbf{p}$  to produce a level set function  $\phi$  that produces a contour in the image

It can thus be seen that the level set function  $\phi$  can be completely described using the parameters  $\mathbf{p} = [a \ b \ h \ \theta]^T$  given the shape prior (hence the “parametric” in the method’s name).

To evolve the level set function,  $\mathbf{p}$  is manipulated in a manner that decreases a predetermined energy functional. This is done using a gradient decent method:

$$\mathbf{p}^{t+1} = \mathbf{p}^t - \alpha_p \nabla \mathbf{p} E \quad 5-3$$

Where  $\mathbf{p}^t$  and  $\mathbf{p}^{t+1}$  are the current and new values of  $\mathbf{p}$  respectively.  $\nabla \mathbf{p} E$  is the gradient of the energy with respect to  $\mathbf{p}$  and  $\alpha_p$  is a step-size parameter controlling the speed of evolution.

This evolution minimises the energy functional  $E$  by moving  $\mathbf{p}$  (i.e.  $a, b, h, \theta$ ) in a direction of decreasing energy. This is made clearer by expanding  $\mathbf{p}$  into its various parameters:

$$\begin{aligned} a^{t+1} &= a^t - \alpha_a \nabla a E \\ b^{t+1} &= b^t - \alpha_b \nabla b E \\ h^{t+1} &= h^t - \alpha_h \nabla h E \\ \theta^{t+1} &= \theta^t - \alpha_\theta \nabla \theta E \end{aligned} \tag{5-4}$$

Where  $\{\alpha_a, \alpha_b, \alpha_h, \alpha_\theta\}$  are step-size parameters for each of the parameters, and  $\{\nabla a E, \nabla b E, \nabla h E, \nabla \theta E\}$  are the gradients of the energy with respect to each of the parameters. As this is a gradient decent method, with each iteration the value of the pose parameters are adjusted in a direction of *decreasing* energy gradient. This moves them (and thus  $\phi$ ) towards values where the functional is minimised. These equations are used iteratively until a predetermined number of iterations have been reached, or the energy no longer decreases with each new iteration.

### 5.3.2 Modifications to Tsai et al's Work

The gradient terms  $\{\nabla a E, \nabla b E, \nabla h E, \nabla \theta E\}$  in equation 5-4 are derived in [10] by symbolically differentiating the energy functional with respect to each of the parameters. This process is mathematically complex and undesirable. The modification to Tsai et al's work is a way to *estimate* these gradient terms thus avoiding the difficult derivation.

Provided the energy functional is differentiable with respect to its input parameters, it is possible to estimate the gradient with respect to each parameter using a numerical method known as the *central difference approximation* thereby avoiding mathematically complex differentiation. Generally, the central difference approximation [51] of a function  $f$ 's gradient is:

$$f'(x) \approx \frac{f(x + \varepsilon) - f(x - \varepsilon)}{2\varepsilon} \tag{5-5}$$

Where  $\varepsilon$  is a small change in the input  $x$ . For level set evolution, the following notation is introduced:

- $\phi_{a,b,h,\theta}$  is the level set function defined by  $a, b, h, \theta$ .
- $E(\phi_{a,b,h,\theta})$  is the output of energy functional calculated using  $\phi_{a,b,h,\theta}$ .

The gradient term  $\nabla a E$ , for example, can then be approximated using the central difference approximation as:

$$\nabla a E \approx \frac{E(\phi_{a+\varepsilon, b, h, \theta}) - E(\phi_{a-\varepsilon, b, h, \theta})}{2\varepsilon} \quad 5-6$$

In this case the numerator is calculated by displacing the original contour defined by  $\phi_{a, b, h, \theta}$  to the left and right in the frame by distance  $\varepsilon$  and finding the difference in energies associated with each contour. Thinking of the process this way may assist the reader in visualising the process of minimising the energy functional with respect to the horizontal displacement pose parameter  $a$ .

This can similarly be repeated for the remaining gradient terms  $\{\nabla b E, \nabla h E, \nabla \theta E\}$ . These are then used to evolve the pose parameters normally according to equation 5-4.

Gradient estimation schemes are in fact more resource intensive than calculation from symbolically derived gradients as they require recalculation of two new level set functions and associated energy functionals every time gradient is estimated. That being said, their main benefit is that any arbitrary energy functional can be plugged into the algorithm without the need for complex symbolic derivations.

### 5.3.3 Implementation Details

To give the insight into the specifics of how the level set segmentation is implemented in code, details are presented in Algorithm 1 which shows the function that controls evolution of the abovementioned parameters.

```

function [energy, outPhi, outP] = LSEvolution(Im,inPhi,inP,nIter,alpha,eps)
    P = inP

    for i = 1 to nIter
        P = OptimParameter(Im,inPhi,P,1, alpha(1), eps(1))
        P = OptimParameter(Im,inPhi,P,2, alpha(2), eps(2))
        P = OptimParameter(Im,inPhi,P,3, alpha(3), eps(3))
        P = OptimParameter(Im,inPhi,P,4, alpha(4), eps(4))
    end
    outPhi = TransformPhi(inPhi,P)
    energy = EvalEnergy(Im,outPhi)
    outP = P
end

```

**Algorithm 1 - Optimisation of transformation parameters for level set evolution.** *The **LSEvolution** function takes input variables **Im** (the image), **inPhi** (the starting level set function), **inP** (the starting pose parameter vector), **nIter** (the number of evolution iterations) and **alpha** and **eps** vectors (step size vectors for gradient decent and gradient estimation). It returns the final level set energy, **outPhi** (the final level set function) and **outP** (the final pose parameter vector)*

The function **LSEvolution** has an input image (**Im**), an initial level set function  $\phi$  (**inPhi**) and pose parameter vector **p** (**inP**) and the number of iterations (**nIter**) required as arguments.

Additionally **alpha** ( $\alpha$ ) and **epsilon/eps** ( $\epsilon$ ) are vectors of parameters that control the step size for evolution and gradient estimation respectively. The individual parameters of these vectors are step-sizes for each individual pose-parameter; the first being for  $a$ , the second for  $b$  and so on.

For every iteration, each component of **p** is evolved individually using the **OptimParameter** function (detailed in Algorithm 2). Once the iterations have stopped, the output  $\phi$  (**outPhi**) is calculated using the transformation from equation 5-2 in **TransformPhi** and outputted along with its associated energy (**energy**) from energy functional (calculated using **EvalEnergy**) and parameter vector **p** (**outP**).

```

function newP = OptimParameter (Im,Phi,oldP,i,alpha,epsilon)
    oldParam = oldP(i)
    paramPlus = oldParam+epsilon
    paramMinus = oldParam-epsilon
    P_Plus = oldP
    P_Minus = oldP
    P_Plus(i) = paramPlus
    P_Minus(i) = paramMinus

    Phi_Plus = TransformPhi (Phi,P_Plus)
    Phi_Minus = TransformPhi (Phi,P_Minus)
    Energy_Plus = EvalEnergy (Im,Phi_Plus)
    Energy_Minus = EvalEnergy (Im,Phi_Minus)
    paramGradient = (Energy_Plus-Energy_Minus)/(2*epsilon)

    newParam = oldParam - alpha*paramGradient
    newP = oldP
    newP(i) = newParam
end

```

**Algorithm 2 – Pseudo-code for the OptimParameter function.** *This function serves to optimise an individual component of the parameter vector  $\mathbf{p}$ . The function takes input variables  $\mathbf{Im}$  (the input image),  $\mathbf{Phi}$  (the level set function),  $\mathbf{oldP}$  (the starting pose parameter vector),  $i$  (the number of the parameter to be optimised) and  $\alpha$  and  $\epsilon$  values for gradient decent and gradient estimation techniques. The output  $\mathbf{newP}$  is a new optimised parameter vector*

The **OptimParameter** function optimises the  $i^{\text{th}}$  parameter component of the vector  $\mathbf{p}$  where  $\mathbf{p}$  contains parameters  $[a, b, h, \theta]$ . For example, when  $i = 2$ ,  $p(i)$  is equivalent to  $b$ .

A small increment ( $\epsilon$ ) is added and subtracted from the original value of the parameter, which is used to transform  $\phi$  to **Phi\_Plus** ( $\phi^+$ ) and **Phi\_Minus** ( $\phi^-$ ) respectively using equation 5-2 in **TransformPhi**. The energy associated with these functions are then obtained using the **EvalEnergy** function and used to calculate an energy gradient using the central difference scheme shown in equation 5-1. This gradient is subsequently used to evolve the parameter (according one of the equations from 5-4) which is passed back to the **LSEvolution** function in a new parameter vector  $\mathbf{p}$  (**newP**).

The details of how this modified level set method is used in both the object detector and tracker stages are shown in the next two sections.

## 5.4 Object Detector

Before the system is able to track respective objects of interest in the image, they first must be detected. This section covers explicit details of each of the subsystems that object detection stage



is comprised of. These include the pre-filtering stage, the background subtraction stage, the post-filtering stage and finally the level set filtering stage.

#### 5.4.1 Pre-Filter

To remove possible noise in the image before it is sent to the background subtraction stage, a pre-filter may be used. Two well-known pre-filters known to reduce noise have been proposed: The 3x3 Gaussian filter and the 3x3 Median filter. The final choice of pre-filter is addressed in Chapter 6.

##### 5.4.1.1 Gaussian Filter

Gaussian blurring convolves the image with a kernel that represents the shape of a Gaussian hump (shown in Figure 5-9). Convolution involves moving the kernel over pixels in the image, multiplying underlying pixel values with corresponding kernel values and replacing the central-pixel's value with the sum of these products.

This filter is used to reduce noise and detail in the image, but more importantly it is able to enhance image structures at various scales. A 3x3 pixel kernel was chosen with Gaussian variance  $\sigma^2 = 1$  to remove detail of smaller objects in the image.

0.1019	0.1154	0.1019
0.1154	0.1308	0.1154
0.1019	0.1154	0.1019

Figure 5-9 - 3x3 Gaussian kernel with  $\sigma^2 = 1$

##### 5.4.1.2 Median Filter

For each pixel in the image the median filter replaces its value with the median of a neighbourhood of pixels around it. Median filters are popular as they are able filter out certain types of noise (impulse noise to be precise) with less blurring than smoothing filters such as the Gaussian filter [17]. A 3x3 median filter is proposed as a possible pre-filter for the system.

### 5.4.2 Background Subtraction

Once the image has been filtered, the system detects objects using a background modelling and subtraction algorithm. By producing a model of the background and subtracting it from the image, it is expected that what remains will be objects of interest. As stated previously in Chapter 3, this requires the objects to be *moving* in the sequence as stationary ones will become part of the background model.

The background model used is based on Elgammal and Duraiswami's work [32] that use KDE to model the background. Using the previous  $L$  values of a particular pixel values  $\{I_{t-L}, I_{t-L+1}, \dots, I_{t-1}\}$  the probability that the next pixel value  $I_t$  has a value  $x$  is estimated as:

$$f(p) = \frac{1}{Lh} \sum_{i=t-L}^{t-1} K\left(\frac{p - I_i}{L}\right) \quad 5-7$$

Where  $K$  is the kernel with bandwidth  $h$  and can be one of a number of different functions. The Gaussian kernel was used:

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-u^2}{2}\right) \quad 5-8$$

While Silverman [34] shows that the best choice for  $h$  for a Gaussians Kernel is:

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \quad 5-9$$

where  $\hat{\sigma}$  is the standard deviation of the data, in this case the previous  $L$  values of the pixel. When a new pixel value  $I_t$  is observed, the probability of its value is calculated from this density estimate.

A *high probability of observation* would indicate that the given pixel is *likely a part of the background* whereas a low probability would indicate a foreground pixel as described in section 3.3. The background subtraction output  $BS_t$  is thus:

$$BS_t(x, y) = \begin{cases} 1 & \text{if } f_{t,(x,y)}(I_t(x, y)) > Th \\ 0 & \text{otherwise} \end{cases} \quad 5-10$$

Where  $Th$  is a predefined threshold.

This model can be modified in a number of ways to better suit the application of background subtraction. It is obvious that more recent pixel values of the pixel are more relevant to the density estimation. For kernel density estimation with time-series data, Harvey and Oryshchenko [52] suggest using a weighting scheme such that:

$$f(p) = \frac{1}{h} \sum_{i=t-L}^{t-1} K\left(\frac{p - I_i}{L}\right) * \omega_i \quad 5-11$$

where  $\omega_i$  is the weight for the  $i^{th}$  kernel. Here  $\sum_{i=1}^n \omega_i = 1$ . In order to weigh more recently viewed pixel values higher, the following weighting scheme was chosen:

$$\omega_i = \frac{i}{n} \quad 5-12$$

Such that the weighting increases linearly with  $i$ . This will ensure that the most *recent* pixel value will have the *highest weight*.

A further modification to the system is the use of frame spacing. Rather than using the entire set of previous  $L$  pixel values, a buffer of  $n$  values is created from pixels spaced  $s$  frames apart:

$$\{I_{t-L}, I_{t-L+1}, \dots, I_{t-1}\} \rightarrow \{I_{t-ns}, I_{t-(n-1)s}, \dots, I_{t-2s}, I_{t-s}\} \quad 5-13$$

Take the example of a sequence of a fairly slowly moving object (such as a ship) captured using a high frame-rate camera. Suppose a buffer of 50 previous frames is used to build a background model. The slow speed of the object combined with the high frame rate of the camera would probably result in very little object movement for these frames, resulting in most of the object

becoming part of the background model. By using frames that are spaced apart, the resultant background model is less likely to include the object because it will have moved over these frames. The disadvantage of this method is that this places an upper limit on how fast an object may be moving so that it is not missed by the spaced frames.

This is implemented in the decision block in Figure 5-4, where frames are only sent to the object detection stage at fixed intervals. Obviously this also means the background subtraction stage is not able to provide a corresponding output for every input frame. This, however, is acceptable as the object tracker stage keeps track of objects for every frame after detection.

Pseudo-code of the background subtraction system is shown in Algorithm 3.

```

function outputFrame = KDEBacksub(currentFrame,threshold)
    for each pixel in currentFrame
        pixelPast = buffer(PixelCoordinate);
        probability = KDE(pixelPast,pixelValue)
        if probability < threshold
            outputFrame(pixelCoordinates) = 1
        else
            outputFrame(pixelCoordinates) = 0
        end
    end
    Delete oldest frame in buffer
    Add currentFrame as most previous frame in buffer
end

function probability = KDE(pixelPast, pixelValue)
    probability = 0;
    sigma = Standard Deviation of pixelPast
    h = calculated from equation 5-9
    n = bufferSize
    Weights = Linear space from 1 to n
    Weights = Weights / Sum(Weights) %Normalise Weights

    for i = 1 to n
        y = Kernel((pixelValue-pixelPast(i))/h)*Weights(i)
        probability = probability + y
    end
    probability = probability / h
end

function [y] = Kernel(x)
    y = 1/sqrt(2*pi)
    y = y*exp((-x.^2)/2)
end

```

**Algorithm 3 – Pseudo-code for KDE-Based Background Subtraction.** The **KDEBacksub** function takes the current frame and the threshold from equation 5-10 as inputs and returns the background subtracted frame in **outputFrame**. The **KDE** function takes in a pixel's past values and its present value and returns a probability. The **Kernel** function is a Gaussian kernel as in equation 5-8

The **KDEBacksub** function applies this background subtraction to each pixel in the image. For each pixel in a new frame, the **KDE** function produces an estimate of a probability using its past (obtained from the buffer) and its current value. It does this using the above formula shown in equation 5-11 which uses the kernel from equation 5-8 (in the **Kernel** function). The output is produced by thresholding the calculated probability as shown in equation 5-10. Once the output has been calculated the oldest frame in the buffer is deleted and the input frame is added to it. This keeps the buffer at a constant size of  $n$  frames.

### 5.4.3 Post-Filtering

Post-filtering is used in this dissertation to describe filtering of the output image of background subtraction. A major problem with the use of background subtraction algorithms alone for maritime surveillance is the motion of the sea. Although kernel density estimation would be able to filter out pixels which oscillate between two values, it still would classify a wave moving across the image, for example, as legitimate motion.

This is illustrated for an example image shown in Figure 5-10. Figure 5-11 is the raw output from a background subtraction algorithm on this sequence. Aside from the red blobs in the image, the majority of detected pixels are considered false positives from waves.



Figure 5-10 – Example frame from an image sequence

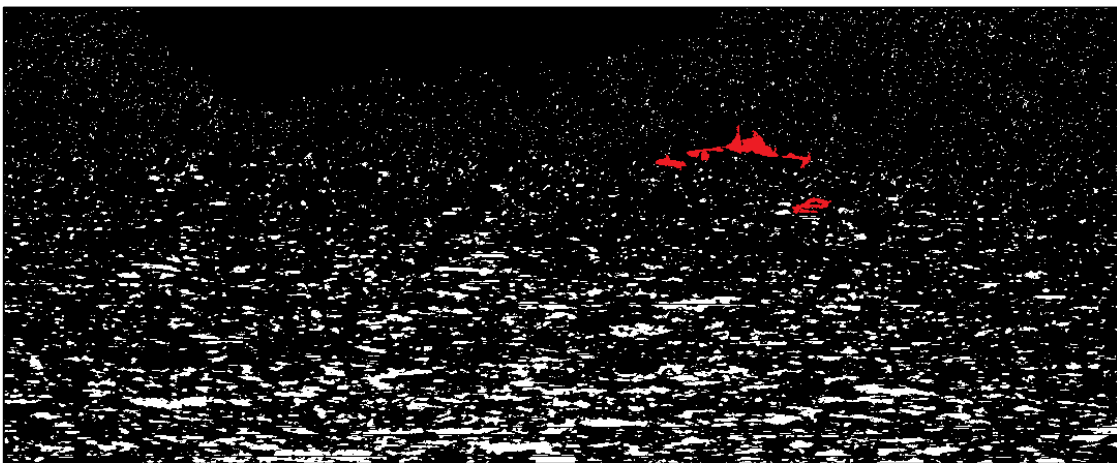


Figure 5-11 - Example raw output of background subtraction. *Pixels resulting from motion of objects of interest have been highlighted in red*

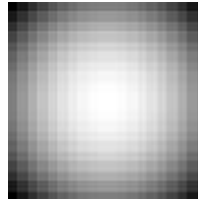
It is for this reason that the binary image is filtered after background subtraction. This subsection details a number of different filters that can be used to remove these unwanted white pixels while keeping the desired ones. The decision of which post-filter to use is addressed in Chapter 6.

#### **5.4.3.1 Motion Persistence Filtering**

Motion persistence filtering is a novel method introduced by this work that attempts to remove white pixels that only appear in a few background subtracted frames. The logic behind motion persistence filtering is that while waves will produce legitimate motion pixels in a background subtraction algorithm, unlike those of ships, this motion is short-lived and may last merely over a few frames. The motion of ships, however, is more “persistent” and thus present in more frames.

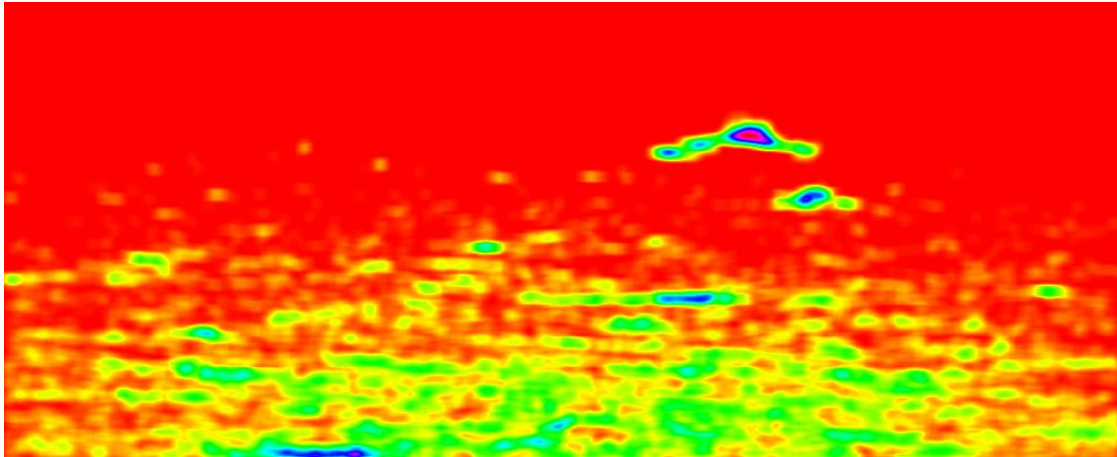
Assuming an input set of background-subtracted images  $\{B_1, B_2, \dots, B_t\}$ , this filter operates in a similar fashion to a two dimensional kernel density estimator:

For every white pixel in an image, a two-dimensional Gaussian kernel (such as the one in Figure 5-12) is placed (centred) over the pixel and its surrounding neighbours. The Gaussian kernel is the same as that described for KDE background subtraction, except projected into two dimensions. The bandwidth of each Gaussian is set to be the distance to the nearest white pixel in the image.



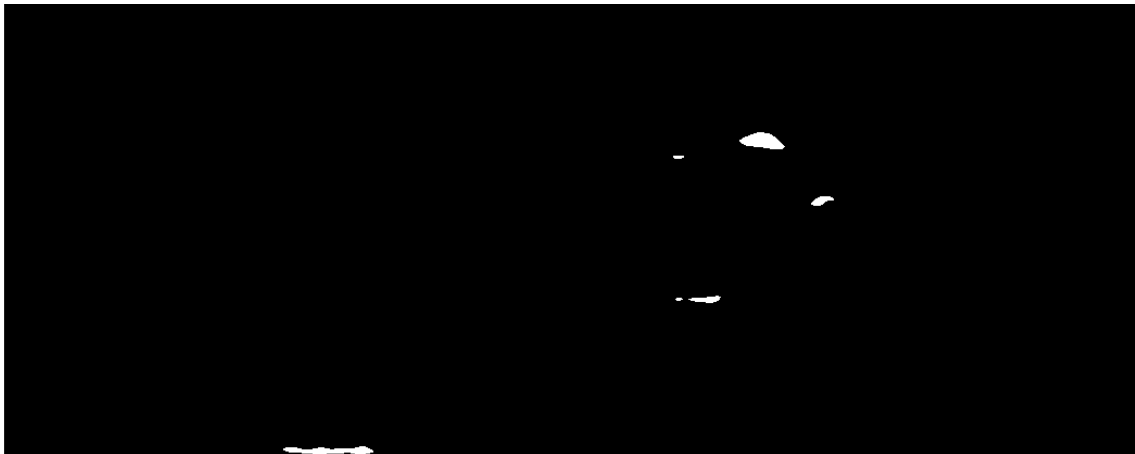
**Figure 5-12 – Two-dimensional Gaussian kernel**

The Gaussian kernels across all pixels in all  $t$  motion images are added up to produce a two-dimensional distribution or heat-map as in the example shown in Figure 5-13. This heat map shows areas of high persistent motion in blue and violet, while green, yellow and red areas represent low densities. The assumption is that areas which consistently contain motion pixels across all frames will have much higher densities than those that contain fleeting motion.



**Figure 5-13 - Example motion persistence heat-map.** *This heat-map shows areas of high persistent motion in blue and violet, while green, yellow and red areas represent low densities*

This image is thresholded to produce a binary image. Figure 5-14 shows the result of thresholding the motion persistence image in Figure 5-13.



**Figure 5-14 - Thresholded version of Figure 5-13.**

The algorithm proceeds to find pixels in the most recent background-subtracted frame  $B_t$  that are in these high persistence areas. The result of filtering Figure 5-11 (the raw output of background subtraction) with this method is shown in Figure 5-15. Of the five remaining blobs that are left, only two belong to that of waves.





Figure 5-15 – Superposition of motion image (red contours) on original frame

Algorithm 4 shows pseudo-code of the motion persistence algorithm. For every white pixel in the binary output of the background subtraction stage, the **FilterMotion** function places a kernel in the neighbourhood around the location of this pixel in the **densityFrame**. After all the frames have been processed, the density frame is normalised to ensure the maximum probability at any given pixel is 1. As detailed above, the system then finds components in the input binary image that are connected to high density areas of **densityFrame**.

```

function outputFrame = FilterMotion(inputFrame,buffer,threshold)
    densityFrame = 0

    Shift buffer
    Add inputFrame to buffer
    K = Two dimensional Gaussian kernel matrix (Fig. 5-12)
    kernelCount = 0

    for each frame in buffer
        currentFrame = current buffer frame
        for each pixel in currentFrame
            if currentFrame (pixelCoordinates) == 1
                kernelCount = kernelCount + 1
                For pixels in square around pixelCoordinates
                    densityFrame = densityFrame + K
                end
            end
        end
    end
    densityFrame = densityFrame / kernelCount
    threshDensity = densityFrame > threshold
    outputFrame = Pixels in inputFrame connected to those in threshDensity
end

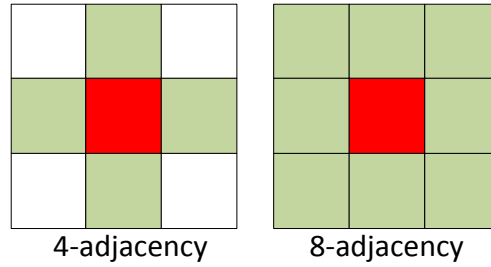
```

**Algorithm 4 - Pseudocode for Motion Persistence Algorithm.** The *FilterMotion* function takes the input frame, a **buffer** of previous frames and a **threshold** as inputs. The threshold is used as described to determine if pixels consist of persistent motion or not. The **densityFrame** variable is a 2D kernel density estimate which is thresholded using the threshold input. Pixels of the input frame that are connected to the thresholded areas of **threshDensity** are output in **outputFrame**

#### 5.4.3.2 Fixed Threshold Connected-Component Filtering

Connected-component filtering is a low-level image processing technique that removes connected regions (or blobs) from a binary image depending on some criteria.

In a binary image, two pixels are said to be connected if they are neighbours and have the same value (0 or 1) [17]. There are numerous criteria that determine if pixels are neighbours (or adjacent). The most common types of adjacency are 4-adjacency or 8-adjacency as illustrated in Figure 5-16.



**Figure 5-16 – Meaning of 4- or 8-adjacency.** For a particular pixel (shown in red) 4- adjacent pixels lie above, below, left or right (shown in green), while 8-adjacent pixels lie in all surrounding pixels

Pixels that are 4-adjacent to a particular pixel lie directly above, below, to the left and to the right of it while 8-adjacent pixels lie in all surrounding pixels. Connected *regions* in an image may be defined using the following description:

“Let  $S$  represent a subset of pixels in an image. Two pixels  $p$  and  $q$  are said to be *connected* in  $S$  if there exists a path between them consisting entirely of pixels in  $S$ . For any pixel  $p$  in  $S$ , the set of pixels that are connected to it in  $S$  is called a *connected component* of  $S$ . If it has only one connected component, then set  $S$  is called a *connected set*.

Let  $R$  be a subset of pixels in an image. We call  $R$  a region of the image if  $R$  is a connected set.”  
[17]

From inspection of Figure 5-11, one may note that the majority of unwanted binary pixels are part of regions that are smaller in size than those that belong to objects of interest. Thus the first connected-component filtering algorithm that is proposed simply removes regions whose size is below a pre-set threshold ( $Th_{fixed}$ ). The algorithm uses 8-adjacency to determine if pixels are part of regions or not.

#### **5.4.3.3 Variable Threshold Connected-Component Filtering**

Voles and Teal [27] note that because a maritime scene is an outdoor one with considerable depth of field, objects close to the camera are projected near the bottom of the image and thus appear larger than those further away from the camera. The second connected-component filtering algorithm is based around this idea. Here, the threshold for blob-size is no longer a pre-set constant, but a linear function of a region’s y-coordinates:

$$Th_{variable} = M(y) \quad 5-14$$

Algorithm 5 shows a snippet of pseudo-code for the filter, where for each component; a y-coordinate of its centroid is calculated. This is normalised with respect to the image height to allow for various image sizes. The threshold is then calculated from a function whose argument is the normalised coordinate and used to filter that specific component.

```
function outputFrame = FilterCCVar(inputFrame)
    outputFrame = Empty Frame

    for each connected region in inputFrame
        Cy = y-coordinate of centroid of region %Zero is top of image
        SizeY = Height of Frame
        Cy = Cy/SizeY %Normalise the y-coordinate
        threshold = ThreshFunc(Cy)
        if size(currentRegion) > threshold
            outputFrame(regionPixels) = 1
        end
    end
end
```

**Algorithm 5 – Pseudo-code from the variable threshold connected-component filter.** The **FilterCCVar** function takes the input frame as an input. The normalised vertical position (**Cy**) is calculated for each connected region and evaluated using the **ThreshFunc** function to produce a threshold. If the region size is over the given threshold it is kept in the **outputFrame**

Finally, a *closing* operation is applied to the output of each of the connected-component filters using a small 2-pixel wide structuring element. This will hopefully fuse two components that may be part of a single object and fill in any holes left by the background subtraction algorithm.

#### 5.4.3.4 Spatial-Smoothness Filtering

Szpak and Tapamo [8] suggest that methods based on thresholding the area of connected regions as described above are not suitable as targets may be smaller than some waves in the image and thus erroneously removed. While a variable threshold should solve this problem, their suggested method of spatial-smoothness filtering is implemented for comparison.

This technique is built into the proposed single Gaussian background subtraction before pixels are thresholded and thus requires some modification; however the expected behaviour is the same. For a pixel at  $(i, j)$  with probability  $f_{KDE}(I(i, j))$ ,  $\Gamma$  is calculated for a window of  $2r \times 2c$  pixels around it as:

$$\Gamma = \sum_{p=-r}^{p=r} \sum_{q=-c}^{q=c} w_{i+p, j+q} \times f_{KDE}(I(i+p, j+q)) \quad 5-15$$

$\Gamma$  is thus the weighted sum of the input pixel  $(i, j)$  and its neighbours probabilities. This effectively is a smoothing operation before the probability estimates are converted into a binary image in the background subtraction algorithm.

The output of the background subtraction  $BS$  for this pixel is then modified as follows:

$$BS(i, j) = \begin{cases} 1 & \text{if } \Gamma < Th \times 2r \times 2c \\ 0 & \text{otherwise} \end{cases} \quad 5-16$$

Where  $Th$  is the background subtraction threshold normally used. It is multiplied by  $2r$  and  $2c$  to adjust for the sum in equation 5-15. Szpak and Tapamo use a  $3 \times 3$  filter with constant weights with values of 1 and so these parameters will be used for testing.

#### 5.4.4 Level Set Filtering

If the shape of a particular object is known beforehand, it is possible (if the user wishes) to filter the binary image further after post-filtering and detect only a region/blob that matches its shape.

While the motion image can be segmented by a level set function minimizing the Chan-Vese energy as shown in [8], this is an inefficient use of resources for the following reasons:

Firstly, the image is binary rather than consisting of grey-scale regions with poor edges for which the Chan-Vese energy is designed. It would be more fitting to use a model that deals with binary images such as the binary mean model proposed by Yezzi and Tsai [53]:

$$E_{binary} = -\frac{1}{2}(c_1 - c_2)^2 \quad 5-17$$

Where  $c_1$  and  $c_2$  are the mean values for the pixels inside and outside  $\phi$ . It is shown in section 4.4.2 that  $c_1$  and  $c_2$  are calculated as:

$$c_1(\phi) = \frac{\iint_{\Omega} IH(\phi)dxdy}{\iint_{\Omega} H(\phi) dxdy} \quad 5-18$$

$$c_2(\phi) = \frac{\iint_{\Omega} I(1 - H(\phi))dxdy}{\iint_{\Omega} (1 - H(\phi)) dxdy} \quad 5-19$$

This energy tries to separate the image into two regions of homogeneous pixel intensity by maximizing the difference between  $c_1$  and  $c_2$ .

Secondly it is known that in an ideal case the level set contour sits tightly around a white blob. In this case, the ideal values for  $c_1$  and  $c_2$  are known. Specifically ideally  $c_1 = 1$  and  $c_2 = 0$ . The binary mean model in equation 5-17 is thus modified accordingly:

$$E = -\frac{1}{4}[(c_1 - 1)^2 + c_2^2] \quad 5-20$$

The first term of this energy penalises inner mean values ( $c_1$ ) that are not equal to 1, while the second term penalises outer mean values  $c_2$  that are not equal to 0.

This energy is minimised using the modified version of Tsai *et al*'s level set scheme [10] detailed in section 5.3. The segmentation is applied for every connected region or blob in the image in isolation. Naturally the blob most likely to be the object sought is that with the lowest energy.

Algorithm 6 shows pseudo-code for this filtering method, where each component in the image is isolated and segmented using a level set with a single shape prior.

```

function outputFrame = FilterLevelSet(inputFrame,shapePrior,nIter)
    outputFrame = 0

    for i = each connected component in inputFrame
        a = 0
        b = 0
        h = 1
        theta = 0
        Get bounding box of component
        SubInput = Empty image with size equal to a scaled-up bounding box
        SubShape = SubInput
        Place component at centre of SubInput
        Resize ShapePrior till same area of component
        Place ShapePrior at centre of SubShape
        Convert SubShape into SDF
        P = [a, b, h, theta]
        [Energy(i), Phi(i) outP] = LSEvolution(SubInput,SubShape,nIter,P)
    end

    minE = Index of minimum element in Energy
    minPhi = Phi(minE)
    Add minPhi to outputFrame at coordinates of its relevant component
end

```

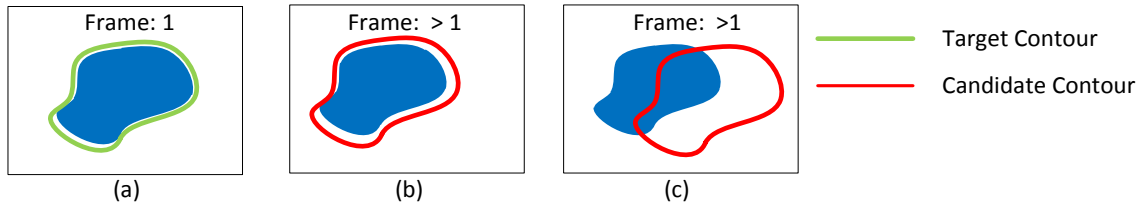
**Algorithm 6 – Pseudo-code for the binary image filtering using a level set with shape prior.** The *FilterLevelSet* function takes the input frame, the shape of the object sought (**shapePrior**) and the number of iterations (**nIter**) as inputs. It goes through each blob in **inputFrame** and returns a level set function centred on the probable location of the object in **outputFrame**

The **FilterLevelSet** function isolates each component by creating a sub-image whose centre is aligned with the centroid of its respective component. A shape prior function is created that matches the size of this image and is used to segment it using the **LSEvolution** function. The **EvalEnergy** function discussed in section 5.3.3 now uses the energy in equation 5-20. The segmenting contour for each blob is saved and that with the lowest energy is placed over its respective blob in the original image.

## 5.5 Object Tracker

Once an object has been detected, its shape and position are known for a single frame. It is necessary to track the object for every frame thereafter. To do this, the object tracker makes use of a single level set function that evolves itself to sit around the object in each frame. The level set function is initialised in the image using the object detector and can come directly from the level set shape filtering stage of the object detector in the form of a single shape (if a prior shape is known and the user wishes), or the binary image at the output of the post-filtering stage in the form of a binary image.

Assuming the level set contour surrounds the object correctly in the first frame after detection, the tracker makes use of pixel information that is within the contour. The initial contour and its inner pixel information in the first frame are henceforth known as the *target contour* and *target model* respectively. For every subsequent frame, the current level set contour (which now probably will not lie around the object) and the information about its inner pixels are known as the *candidate contour* and *candidate model* respectively. By creating an energy functional that penalises deviations of the candidate model from the target model, one is able to force the candidate contour around objects appearing similar to those surrounded by the target contour in the original frame. This is illustrated in Figure 5-17, where energy associated with the contour in (b) will be less than that of the contour in (c) and so (b) is favoured.



**Figure 5-17 - Example of tracking by comparing inner pixel models of contours.** As the target contour sits around the blue object in the first frame (a), the energy associated with the contour in (b) will be lower than that in (c)

This method draws its inspiration from Comaniciu *et al*'s method of tracking using a Gaussian Kernel [24], where an elliptical kernel is used and the internal pixels' histogram is used as a model. Different energy functionals may be created by comparing the target and candidate models in various ways. The various possible functionals are discussed next.

## 5.5.1 Energy Functionals for Tracking

### 5.5.1.1 Histogram

One of the simplest features that can be drawn from the pixels is the histogram. Pixels are put into  $k$  bins where  $h_i^t$  is the number of pixels that fall into the  $i$ th bin for the target  $t$ , and  $h_i^c$  for the candidate  $c$ . The energy is the sum of squared differences of the bins:

$$E = \sum_{i=1}^k (h_i^t - h_i^c)^2 \quad 5-21$$



### 5.5.1.2 Fast Fourier Transform

Frequency information may be utilized to make the feature more invariant to changes in lighting. Given a bounding box around the contour  $M \times N$  pixels in size, a modified FFT is used to only extract frequency information from pixels within the contour:

$$F_{\phi}(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} H(\phi(m, n)) I(m, n) e^{-i2\pi(\frac{xm}{M} + \frac{yn}{N})} \quad 5-22$$

The energy function is then defined as the difference in target and candidate spectra:

$$E = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |F_{\phi_c} - F_{\phi_t}| \quad 5-23$$

### 5.5.1.3 Statistical Descriptors

Statistical descriptors of the target pixels can be calculated. This approach has been used previously in maritime tracking work by Voles and Teal [27]. The following descriptors in Table 5-1 have been modified to suit a level set case, once again for a bounding box around the contour  $M \times N$  pixels in size:

**Table 5-1 - Statistical descriptors for pixels within a level set**

Energy:	$d_1 = \sum_{m=0}^M \sum_{n=0}^N H(\phi(m, n)) \cdot I(m, n)^2$
Entropy:	$d_2 = \sum_{m=0}^M \sum_{n=0}^N H(\phi(m, n)) \cdot I(m, n) \cdot \log(I(m, n))$
Homogeneity:	$d_3 = \sum_{m=0}^M \sum_{n=0}^N \frac{H(\phi(m, n)) \cdot I(m, n)}{1 +  m - n }$
Contrast:	$d_4 = \sum_{m=0}^M \sum_{n=0}^N H(\phi(m, n)) \cdot (m - n)^2 \cdot I(m, n)$

After normalizing with respect to a maximum value, these descriptors can be thought of as vectors that form a basis for a 4D space. The target contour's pixel distribution is then represented as a point within this space  $D^t = [d_1^t, d_2^t, d_3^t, d_4^t]$  and similarly so for a candidate contour  $D^c = [d_1^c, d_2^c, d_3^c, d_4^c]$ . The Euclidean distance between these two points can then be used as the energy functional:

$$E = \sqrt{\sum_{k=1}^4 (d_k^t - d_k^c)^2} \quad 5-24$$

### 5.5.2 Implementation Details

Algorithm 7 shows details of the level set tracking method. For the first frame of detection, the pose parameters **a**, **b**, **h** and **theta** ( $\theta$ ) are initialised and the target contour **PhiT** ( $\phi_t$ ) is obtained. The pixels within this contour in the first frame are saved for later use in the energy functionals.

```

function OutputFrame = LevelSetTrack(InputFrame, ShapePrior, nIter)
    OutputFrame = 0;
    if tracking has just started
        a = 0
        b = 0
        h = 1
        theta = 0
        PhiT = Target Phi obtained from object detector
        Save internal pixels of target Phi for energy functionals
    else
        a, b, h and theta take their values from previous frame
    end

    for i = each connected component in InputFrame
        P = [a b h theta]
        [E, PhiC, P] = LSEvolution(InputFrame, PhiTarget, P, nIter)
    end
    Save a, b, h and theta values for use in next frame
    OutputFrame = Draw candidate Phi's (PhiC) contour on InputFrame
end

```

**Algorithm 7 – Pseudo-code for the level set tracking algorithm.** The *LevelSetTrack* function takes the input frame, the shape of the object tracked (*shapePrior*) and the number of iterations (*nIter*). It returns an *outputFrame* that is the original image with the shape drawn around the object

For every frame thereafter, level set evolution is applied using the **LSEvolution** function described in Algorithm 1. The **OptimParameter** function and specifically the **EvalEnergy** function in

Algorithm 2 now uses one of the above functionals, combined with the previously saved **PhiT** and its internal pixels to evaluate the energy at various candidate contours.

The best candidate contour after evolution is output and drawn on the input frame and the pose parameters are saved for the next frame. This step is important: subsequent frames *do not* start evolution from the initial position of the target contour  $\phi_t$ , but rather from where the contour of the previous frame left off.

### 5.5.3 Normalisation for Rotation/Scale Invariance

Apart from energy functionals for tracking, the second pertinent issue in object tracking is consideration of possible rotation or scale changes of the object being tracked. Evolution using the abovementioned functionals may become erroneous for large differences in scale and rotation between the candidate and target level set functionals. Consider the example case of an object tracked using the histogram-based energy becoming increasingly large as tracking continues. While the distance between target and candidate histograms may initially be small, as the object (and thus candidate contour) grows the size of each of its bins increases which causes an increase in the distance from the target histogram. This distance *may* be lower than its surrounding pixels, however this increase is undesirable.

While an obvious solution may be to normalise the histogram with respect the size of its associated contour, one may apply a generalised normalisation that would apply for any arbitrary functional. This normalisation would also eradicate error associated with changes in rotation for other functionals. Assuming an arbitrary scale parameter  $h$  and rotation parameter  $\theta$  that produces the candidate function  $\phi_c$ , both the current image frame  $I$  and the candidate function  $\phi_c$  are transformed according to:

$$\tilde{\phi}_c(x, y) = \phi_c(\tilde{x}, \tilde{y}) \quad 5-25$$

$$\tilde{I}(x, y) = I(\tilde{x}, \tilde{y}) \quad 5-26$$

Where  $\tilde{x}$  and  $\tilde{y}$  are calculated as:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \begin{bmatrix} 1/h & 0 & 0 \\ 0 & 1/h & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad 5-27$$

This transforms both the candidate function and the pixels it contains to the same scale and rotation as the target function. The transformed function  $\tilde{\phi}_c$  and image  $\tilde{I}$  are then used in the evaluation of the energy functional. It should be emphasised that the values of  $h$  and  $\theta$  do not change and that the original candidate contour and image remain intact: their transformed values are used exclusively for evaluating energy functionals.

## 5.6 Conclusion

This chapter presents details of the system proposed. An overview of the system is initially presented, where each subsystem and its relationship with its surroundings are briefly covered. The incorporation of shape knowledge is based around a level set segmentation method proposed by Tsai *et al* [10] and so explicit details of this work are presented. Furthermore, Tsai *et al*'s work is modified to fit into the system model and these modifications are explained in this chapter. The full details of how this modified level set method fits into the video tracking system and details of its surrounding subsystems are finally presented. The choices of the final sub-systems and their parameters are discussed in Chapter 6.

## **Chapter 6 - Experimentation and Results**

The details of the video tracking system have been extensively discussed; in particular the details of the object detection and object tracking algorithms have been explained. A specification of each of the pertinent modules of functionality has been provided. There are a number of different choices that have to be made regarding each of the various sub-systems which have been described in Chapter 5. For the object detector subsystems: the choice of pre-filter; background subtraction buffer size, spacing and threshold; and post-filter must all be selected. For the object tracker, the optimal energy functional and number of iterations must be considered.

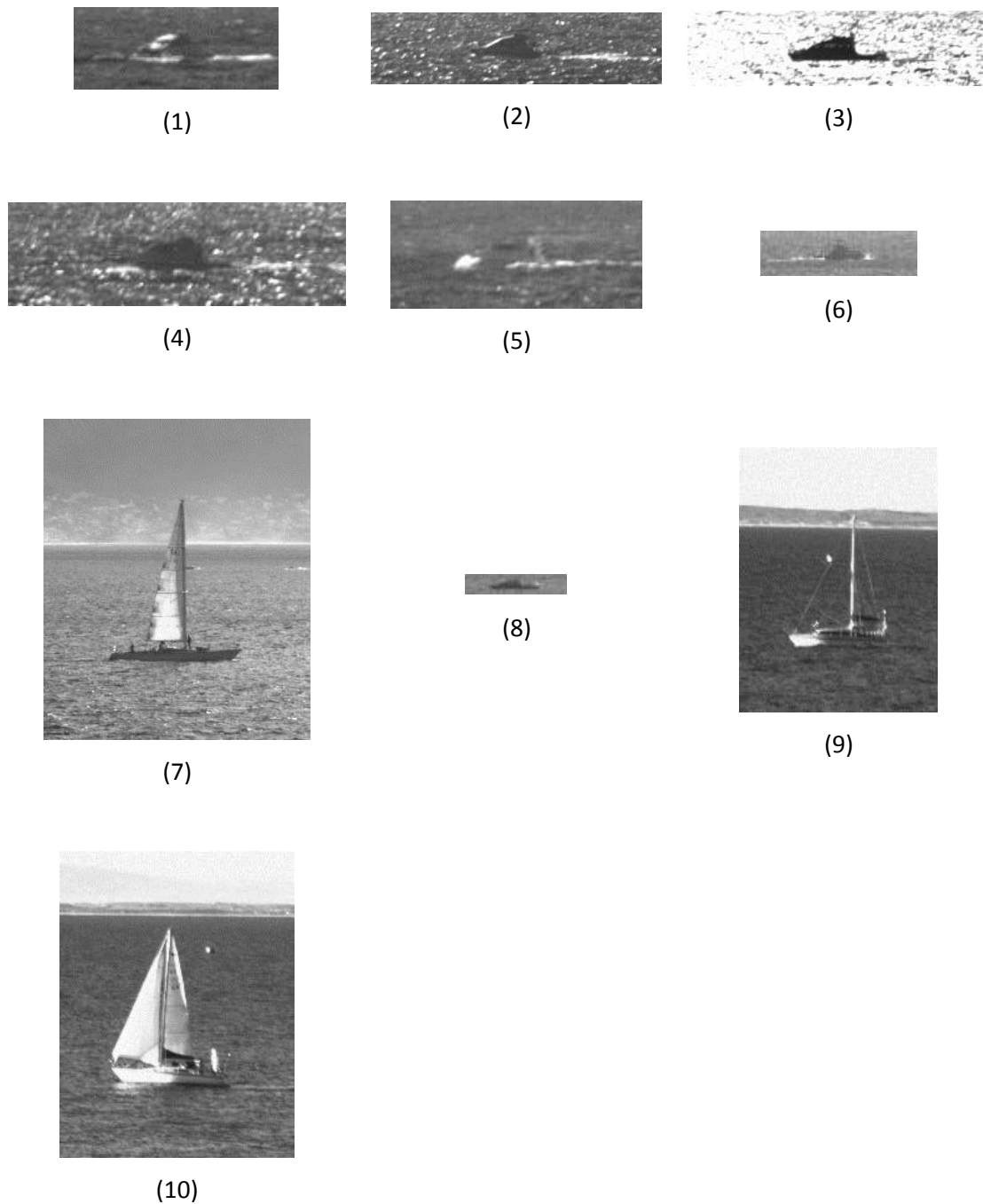
This chapter considers more closely the relevant factors in choosing system's parameters reasonably. In order to justify whether a sub-system has any merit, it is important to measure performance. With the chosen parameters implemented in the video tracker, the system performance is measured and commented upon.

### **6.1 Experimental Conditions**

The system was implemented in MATLAB and tested using a set of 10 maritime sequences obtained from CSIR Defence, Peace, Safety and Security<sup>1</sup>. These sequences include a variety of scenes, weather conditions and maritime objects of interest. A specific target object was chosen for each of the sequences and was used to test both object detection and tracking. The target objects for each of the sequences are shown in Figure 6-1.

---

<sup>1</sup> Council for Scientific and Industrial Research (CSIR) Defence, Peace, Safety and Security - <http://www.csir.co.za/dpss/>



**Figure 6-1 – Target objects as they appear in their original image sequences.** *The sequence from which the object comes is labelled below its respective window*

Table 6-1 details qualitative and comparative descriptions of the target object size and the weather conditions present for each sequence. It is clear that a variety of object sizes and weather conditions are present and so these sequences are a good representative sample to be used in

experimentation. Since the sequences are from a reputable source it is assumed that they are a good representation of scenes likely to be encountered in an active environment.

**Table 6-1 - Qualitative descriptions of target object size and weather for each sequence**

<b>Sequence</b>	<b>Object Size</b>	<b>Weather</b>
<b>1</b>	Medium	Overcast
<b>2</b>	Medium	Glint
<b>3</b>	Medium	Glint
<b>4</b>	Medium	Glint
<b>5</b>	Medium	Overcast
<b>6</b>	Small	Clear
<b>7</b>	Large	Overcast
<b>8</b>	Small	Clear
<b>9</b>	Large	Clear
<b>10</b>	Large	Clear

The object detection stage was tested using sequences 2, 6, 7 and 10 (highlighted in Table 6-1); while the object tracking stage was tested using the entire set. Details of sequence selection are described in their relevant subsections.

## **6.2 Ground Truth Estimation**

Ground truth in an image processing context is loosely defined as a reference image that shows objective reality about a particular situation [54]. In the context of object detection or tracking, this refers to the identification of the pixels in the image that are known with absolute certainty to belong to objects of interest and those that don't. This is inherently important for performance measurement where the system's estimates of these pixels are compared to the actual data.

Unfortunately the sample image sequences used for testing did not have ground truth available and plans to overcome this challenge had to be devised. Obviously labelling a sequence of three-hundred frames or so by hand would not be viable and ground truth had to be estimated.

While Szpak and Tapamo chose to simply hand-label a bounding box around the object in each frame of these sequences [8], ground truth was estimated using the following technique:

For a particular image sequence, the ground truth of the first frame was labelled by hand. For each frame thereafter, the bounding box was hand-labelled around the object. The ground truth from the first frame was then translated and resized in order to fit the bounding box for the current frame.

This is a valid estimation provided the object in question does not rotate with respect to the camera. This is a valid assumption as most objects consist of ships on the ocean. Qualitative inspection of this estimate compared with the image shows it provides a sufficiently accurate estimate of the actual ground truth.

## **6.3 Performance Metrics**

In order to measure performance a set of metrics must be devised. While the majority of metrics detailed here are standard measures of performance, some have been devised or modified in order to measure behaviour unique to the system. These metrics are now described and where they are applied during performance measurement is explained.

### **6.3.1 Object Detection**

The object detection algorithm is a form of a classification task, where pixels are either classified as belonging to a maritime object or not. The detector's output is a binary image with pixels that are classified as objects being labelled as 1.

Given the actual classification of pixels (ground truth) and the output of the system, four outcomes are possible as detailed in Figure 6-2. Outcomes where the system agrees with the actual data are labelled true positives (TP) or true negatives (TN) depending on whether the pixel belongs to an object or not. If the system incorrectly labels a pixel as an object when in actuality there isn't one there this is called a false positive (FP), while a false negative (FN) is a case where an object is present but the system fails to detect it.



		PREDICTED	
		NEGATIVE	POSITIVE
ACTUAL	NEGATIVE	<b>tn – True Negative</b> Correct rejections	<b>fp – False Positive</b> False alarms Type I Error
	POSITIVE	<b>fn – False Negative</b> Misses, Overlooked Danger Type II Error	<b>tp – True Positive</b> Hits

Figure 6-2 - Table of relations between actual and predicted output

For a classification task, *Precision* is defined as, given the actual classifications of particular subjects, the proportion of cases where the subject was classified as positive and were actually the case [55]. Precision is calculated as:

$$Precision = \frac{TP}{TP + FP} \quad 6-1$$

Where TP and FP are true positive and false positive totals for the entire image. *Recall* is defined as the proportion of subjects which were actually positive and were classified as such [55]. Recall is calculated as:

$$Recall = \frac{TP}{TP + FN} \quad 6-2$$

Where TP and FN are true positive and false negative totals for the entire image. Hripcsak and Rothschild [55] define the *F-score*, which is a harmonic mean of the two metrics:

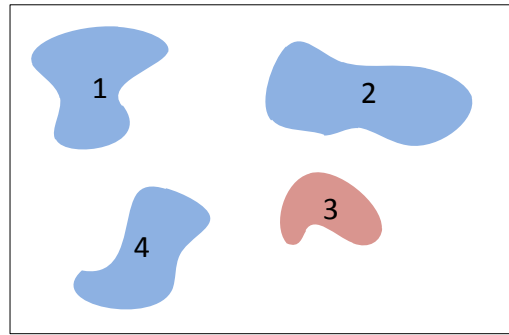
$$F_{\beta} = \frac{(1 + \beta^2) * Recall * Precision}{\beta^2 * Precision + Recall} \quad 6-3$$

Where  $\beta$  is a parameter that allows one to weight Precision or Recall more heavily. Szpak and Tapamo [8] note that for a surveillance system the reduction of false negatives is top priority. It is preferable to have a system give false alarms than falsely ignore incoming vessels which may potentially pose a threat. For this reason Recall was weighted twice as much as Precision by setting  $\beta = 2$ . This  $F_2$ -score can be used to test both the output of the background subtraction algorithm and the post-filter.

To test the accuracy of level set filtering, the output of the post-filtering stage was segmented using binary level set shape prior segmentation for each sequence. For an input binary image with  $n$  blobs, and assuming energy is positive at all times the segmentation proficiency score is defined as:

$$P = \frac{\overbrace{\argmin_{i \neq t \leq n} E(i)}}{E(t)} \quad 6-4$$

Where  $E(x)$  is the final energy obtained from the level set segmentation of blob  $x$  in the image and  $t$  is the index of the blob belonging to the object that is to be found. This score has been devised specifically to measure performance of the level set filtering algorithm and does not feature in any previous literature. It essentially measures the contrast between the energy associated with segmentation of the actual object and the blob with the lowest energy of the remaining blobs. The significance of this score can be illustrated with an example shown in Figure 6-3 where blob 3 corresponds to the blob of the desired object. Each of the blobs has its own associated energy and so the numerator in equation 6-4 will consist of the lowest energy of blobs 1, 2 and 4. The denominator will consist of the energy associated with blob 3.



**Figure 6-3 - Example image consisting of  $n$  different blobs**

If blob 3 has the lowest energy of all blobs in the image,  $P$  will be greater than 1 indicating a correct classification, however if another blob has a lower energy than 3, the level set filtering algorithm will incorrectly classify this as the object and  $P$  will be less than 1. In this way,  $P$  is used to measure the success of the level set filtering algorithm.

If the object is successfully detected, the quality of its segmentation is measured. Krishnaveni and Radha [56] suggest using Dice's coefficient [57] as a method of performance evaluation for level set methods:

$$s = \frac{2 (A \cap B)}{A + B} \quad 6-5$$

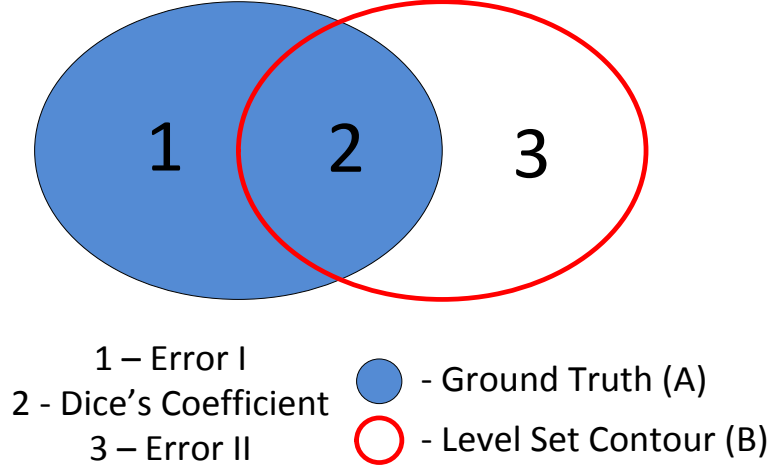
Where  $A$  and  $B$  are the ground truth and resultant segmentation regions respectively and  $s$  varies from 0 to 1 depending on the proportion of pixels shared between the ground truth and the segmenting contour. While this measure is a good rule of thumb, it lacks the measure of separate error types. Type I error is defined as the proportion of ground truth pixels not covered by the segmenting contour:

$$E_I = \frac{A - (A \cap B)}{A} \quad 6-6$$

And Type II error is defined as the portion of segmentation region pixels that cover regions that are not part of the ground truth:

$$E_{II} = \frac{B - (A \cap B)}{B} \quad 6-7$$

A physical analogy of each metric is shown Figure 6-4. Error I is derived from region A, which consists of the portion of the ground truth object not covered the level set contour. Dice's coefficient is derived from the intersection of the level set contour and the ground truth object, and error II is derived from the portion of the inside of the level set contour that does not segment the ground truth object.



**Figure 6-4 - Physical analogy of segmentation metrics.** For two intersecting regions, sub regions can be classified as contributions to one of three possible metrics

### 6.3.2 Object Tracking

In order to evaluate the performance of an object tracker for a particular sequence of images two basic metrics are commonly used:

- Of all the frames in the sequence, for how many was tracking considered to be successful?
- For those frames that it was considered successful, how accurate was the tracking?

The tracker detection rate (TRDR) is the average number of frames an object is successfully tracked and is defined by Porikli and Bashir [58], as:

$$TRDR = \frac{TPF}{TG} \quad 6-8$$

Where  $TPF$  is the number of frames the system contour overlaps the ground truth object.  $TG$  is the number of frames in which the ground truth object is present. There are different strategies to test if object overlap occurs, the simplest of which is to test if the system contour's centroid lies within the ground truth object's bounding box.

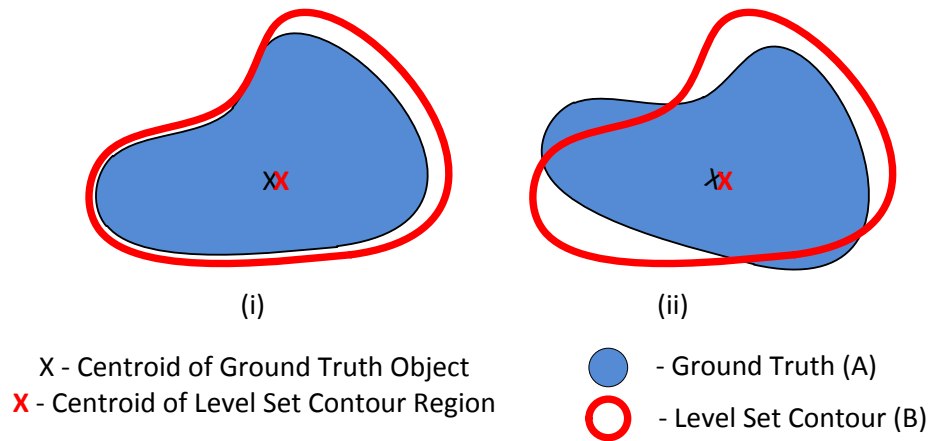
To measure the degree of success for a tracked object, the object tracking error (OTE) [58] is defined as:

$$OTE = \frac{1}{TPF} \sum_{i=1}^{TG} Dist(p_i^{GT}, p_i^{Sys}) * G(GT_i, Sys_i) \quad 6-9$$

Where  $Dist()$  measures the distance between are the centroids for the ground truth object  $p_i^{GT}$  and system contour  $p_i^{sys}$  in the  $i^{th}$  frame respectively.  $G(GT, Sys)$  is an overlap function defined as:

$$G(GT, Sys) = \begin{cases} 1 & \text{if } GT \text{ and } Sys \text{ overlap} \\ 0 & \text{if not} \end{cases} \quad \text{6-10}$$

Unlike most other trackers level set-based methods provide an estimate of the ground truth objects outline for each frame. Figure 6-5 shows the problem of evaluating level set trackers' performance simply by object tracking errors. Here two potential tracking outputs might have a similar object tracking error as the distance between the two regions' centroids remains the same, yet the contour in (i) is clearly a better fit than in (ii).



**Figure 6-5 – Comparison of two situations that would give a similar object tracking error. Despite having similar errors, (i) would be a more desirable output.**

For this reason the contour tracking error (CTE) is used in conjunction with the OTE and defined as:

$$CTE = \frac{1}{TP} \sum_{i=1}^{TP} \frac{1}{(1 + \alpha)} [\alpha E_I(GT_i, Sys_i) + E_{II}(GT_i, Sys_i)] * G(GT_i, Sys_i) \quad \mathbf{6-11}$$

Where  $E_I$  and  $E_{II}$  are the segmentation errors described in equation 6-6 and equation 6-7 respectively and  $G(GT, Sys)$  is the overlap function defined above. For the same reasons as the choice of  $\beta$  for the  $F_2$  score,  $\alpha$  is set to 2. The contour tracking error simply is thus the mean contour error (defined as a weighted sum  $E_I$  and  $E_{II}$ ) over frames that were successfully tracked.

## 6.4 Optimisation

The methods described in the previous chapter are dependent on a number of user-given parameters. The choice of these parameters is not always clear from inspection and so algorithms were devised to find an optimal parameter value. This is achieved by setting up a fitness or cost function and varying parameters until an optimum of this function is found. The function is designed to penalise undesirable behaviour and reward desirable characteristics in each algorithm.

This chapter derives optimal user-input parameters where they are required in the modules described in the previous chapter. These parameters are then implemented in the video tracking system and performance of this system is measured in the next section.

Realistically, a large number of datasets must be used to find *optimal* parameters. Since only a small dataset is available for testing, the word *optimal* is used rather loosely. Therefore, while the same optimisation processes that are executed in this work could be applied to a larger dataset, it is recognised that the derivation of the parameters here is in no way rigorous. Future work may pursue a more thorough optimisation but for the purposes of this work it is sufficient to present the concept and some sample parameter values.

### 6.4.1 Object Detection

Figure 6-1 shows the windows taken from the 10 original image sequences, each containing their respective objects. While the common strategy of splitting a dataset into 2/3 for training and 1/3 for testing is shown to work well for reasonably sized datasets (over 100 cases [59]), this was used for the system dataset despite its small size. Sequences 2, 6, 7 and 10 were chosen as a test set while the remaining sequences were used for training. The test sequences are a good representative sample of the entire set as they present a variety of object sizes and weather conditions as shown in Table 6-1.

The optimisation for each subsystem of the object detector is discussed.

#### 6.4.1.1 Pre-Filtering

Neither of the two pre-filters described previously have input parameters that need to be derived – the filters are simply applied in a standard way. Therefore the topic of pre-filters is not discussed again until the next section where the performance of each pre-filter is presented and discussed.

#### 6.4.1.2 Background Subtraction

In order to achieve a more satisfactory result, the background subtraction algorithm must be optimized with respect to:

- The number of frames used for kernel density estimation ( $n$ ).
- The spacing between each frame ( $s$ ).
- The probability threshold used ( $Th$ ).

These variables were adjusted to minimise a cost function that is calculated for a small window around the ground truth object:

$$C(n, s, Th) = \alpha * E_I(GT_{n+1}, BS_{n+1}) + E_{II}(GT_{n+1}, BS_{n+1}) \quad 6-12$$

Where  $BS$  is the output of the background subtraction algorithm and  $\alpha$  is a mixing parameter to allow more emphasis on minimisation of Type I and Type II error. As reduction of false negatives is more important,  $\alpha$  was set to 2.

A brute force search was run to find optimal parameters for each of the above-mentioned training sequences. For every sequence, the values of  $n$ ,  $s$  and  $Th$  were varied and the cost function was measured. Those that resulted in the lowest cost function were considered optimal for each sequence and are shown in Table 6-2.

Table 6-2 - Optimal background subtraction parameters for various sequences

Sequence Number	Optimal $n$	Optimal $s$	Optimal $Th$
1	19	10	0.015
3	19	15	0.005
4	18	14	0.01
5	14	13	0.015
8	8	8	0.01
9	18	5	0.015

The result of using background subtraction using these individual parameters is shown in Figure 6-6.

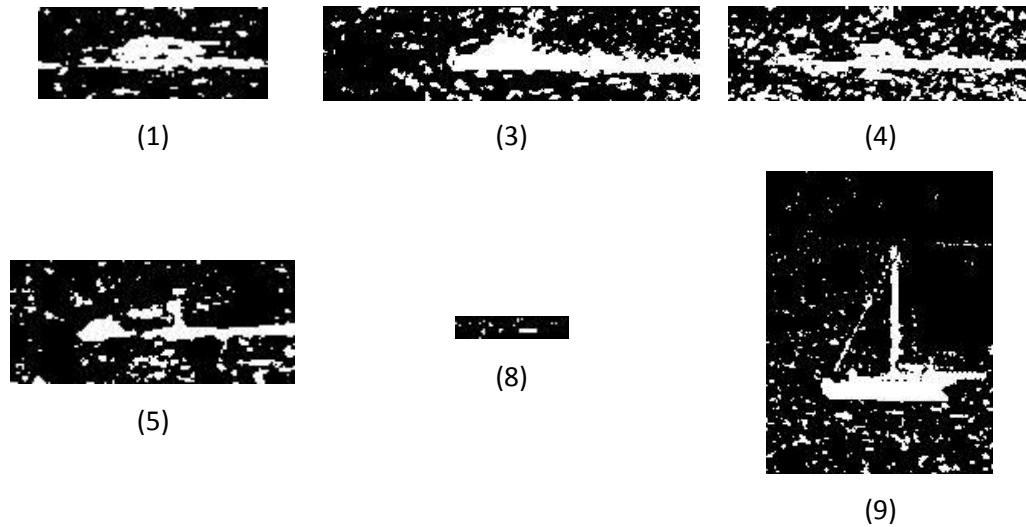


Figure 6-6 - Background subtraction results for optimal individual parameters

The averages of these parameters were used in the final algorithm in anticipation that these would give fairly decent results for across most of the sequences.

Therefore the optimally chosen parameters are:

- $n = 16$
- $s = 11$
- $Th = 0.0117$

#### 6.4.1.3 Post-Filtering

Out of the possible post-filters listed, the fixed threshold connected-component algorithm has an optimal threshold based on region/blob size, the variable threshold algorithm has an optimal threshold function that may be used and the motion persistence filtering algorithm has an optimal number of frames that may be used. The values of these parameters are now determined.

#### Motion-Persistence Filtering

It was empirically decided to use 3 previous background subtracted frames for motion persistence filtering. If background subtraction with frame-spacing is used there should be considerable changes in sea motion across these frames.

#### Fixed Threshold Connected Component Filtering



To find an optimal fixed threshold for fixed threshold connected-component filtering, the area of the smallest ground truth object in each sequence was measured. The algorithm has to filter out regions which are superfluous but still keep those that belong to actual maritime objects. To do this, a lower threshold for blob-size must be selected. Table 6-3 shows the area of the smallest ground truth object in each sequence.

**Table 6-3 - Area of smallest ground truth object in each sequence**

Sequence	Object Area (Total number of pixels)
1	458
3	1047
4	696
5	677
8	70
9	1468

The smallest object, observed in sequence 8, has an area of 70 pixels. To ensure that objects above this size are kept, and allowing 10 pixels for safety: the threshold set at:

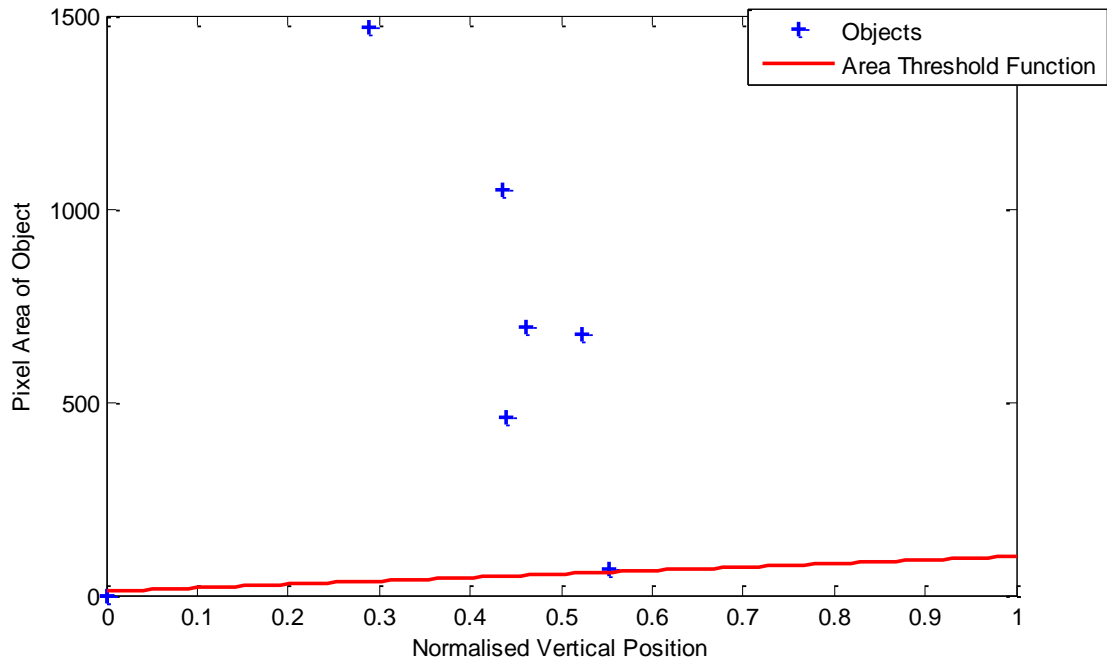
- $Th_{fixed} = 60$

### **Variable Threshold Connected Component Filtering**

For a variable threshold connected-component algorithm, the optimal threshold function must be obtained. Figure 6-7 shows the area of each of the training objects plotted vs. their normalised vertical position. Here 0 represents the top of the image while 1 represents the bottom. The following area threshold function was chosen:

$$M(y) = 90y + 10 \quad \text{6-13}$$

Where  $y$  is the normalised vertical position. This function (plotted in red in Figure 6-7) ensures that blobs near the top of the image need only be over 10 pixels in size to be kept in the image, while blobs at the bottom of the image need to have an area over 100 pixels to be kept.



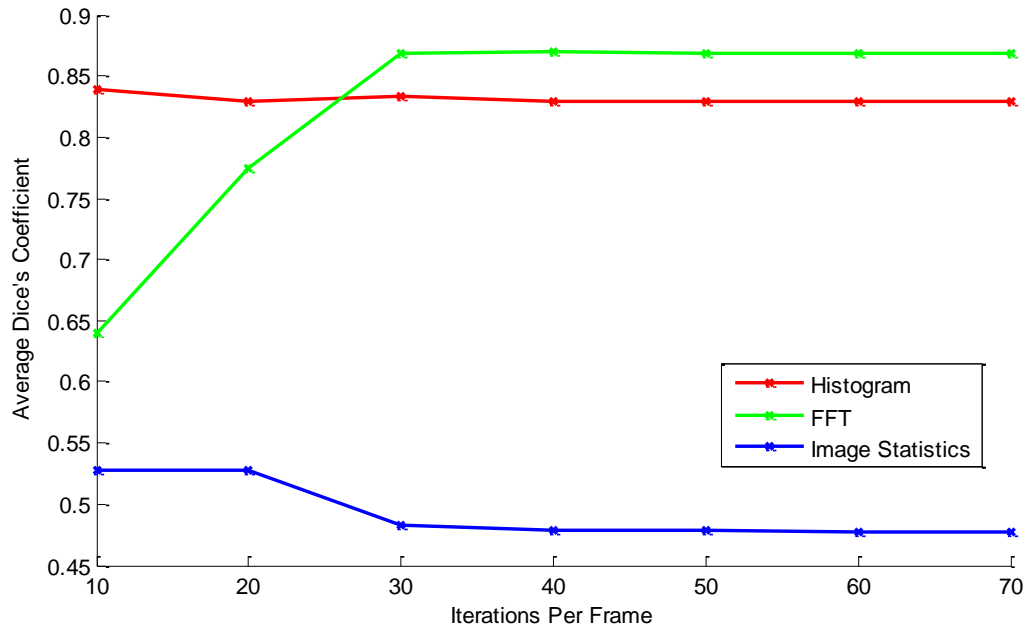
**Figure 6-7 - Plot of ground truth area vs. vertical position in image.** *The chosen threshold function is shown in red*

The large variation in pixel area for each of the objects can be attributed to the various scenes in which they appear. Some sequences are filmed quite far away from the ocean and so objects appear to be very small, while others are filmed very close and produce large objects in the image. If a large amount of data were available from a single static scene a large number of points could be plotted onto Figure 6-7, which would reveal a more specific trend in the increase in object's area with vertical position.

#### 6.4.2 Object Tracking

Section 5.5 explained that the important parameters that need to be selected for object tracking are energy functional and the number of iterations used during evolution of these energy functionals towards the desired minimum point. In this section optimal values for both of these parameters are derived. While the choices of various object detection stages are highly dependent on the sequence target object in question, various object tracking energies described in section 5.5.1 were found to exhibit identical behaviour across all sequences. It is for this reason that the 10 sequences (shown in Figure 6-1 and Table 6-1) were not split into training and testing sets. The tracking algorithm was optimised for a few sequences and tested on all of them.

For each energy term, the algorithm was run on a hundred frames for various iterations per frame. Three randomly-selected sequences were used as a source of these frames. The average Dice's coefficient vs. the number of iterations per frame for each energy term is shown in Figure 6-8.



**Figure 6-8 - Average error vs. Number of Iterations for various energy terms**

Although it has fairly poor results at low iteration counts, the FFT-based energy gives the best results after 30 iterations per frame. Naturally one wishes to keep the number of iterations low to ensure efficiency while keeping them high enough to ensure proper segmentation. When iterations reach 30 the average Dice's coefficient tapers off and so, adding 10 iterations for safety, 40 iterations per frame was chosen. The final parameters of the tracking algorithm are thus:

- Energy used: FFT difference
- Number of iterations per frame: 40

A possible reason for the low performance of the statistical-feature energy functional is most likely due to a lack of functional smoothness with respect to each of the transformation parameters. If the functional does not decrease as the contour gets close to the object, the gradient would not necessarily produce evolution in a favourable direction. This would make finding a global minimum very difficult and cause the candidate contour to deviate from the object.

## 6.5 Object Detection Results

Given the parameters deduced in section 6.4.1, the object detection performance using these parameters can now be investigated. The optimised object detection algorithm was run on the test set of sequences: Sequences 2, 6, 7 and 10. The entire system has been broken into its individual sub-systems which have been tested individually. Sub-systems that have a number of different algorithms at their disposal have been tested using each of these techniques and compared.

### 6.5.1 Pre-Filter and Background Subtraction

The choice of pre-filter has been postponed until now. Figure 6-9 shows  $F_2$ -scores for the output of the background subtraction stage with and without various pre-filtering algorithms. Recall,  $F_2$  refers to the F-score where  $\beta = 2$ . The comparatively low scores for sequences 2 and 6 can be explained by the large amount of glint in sequence 2 (see Figure 6-11) and the small object size in sequence 6 (see Figure 6-12). Despite these low scores, pre-filtering was able to improve scores for every sequence.

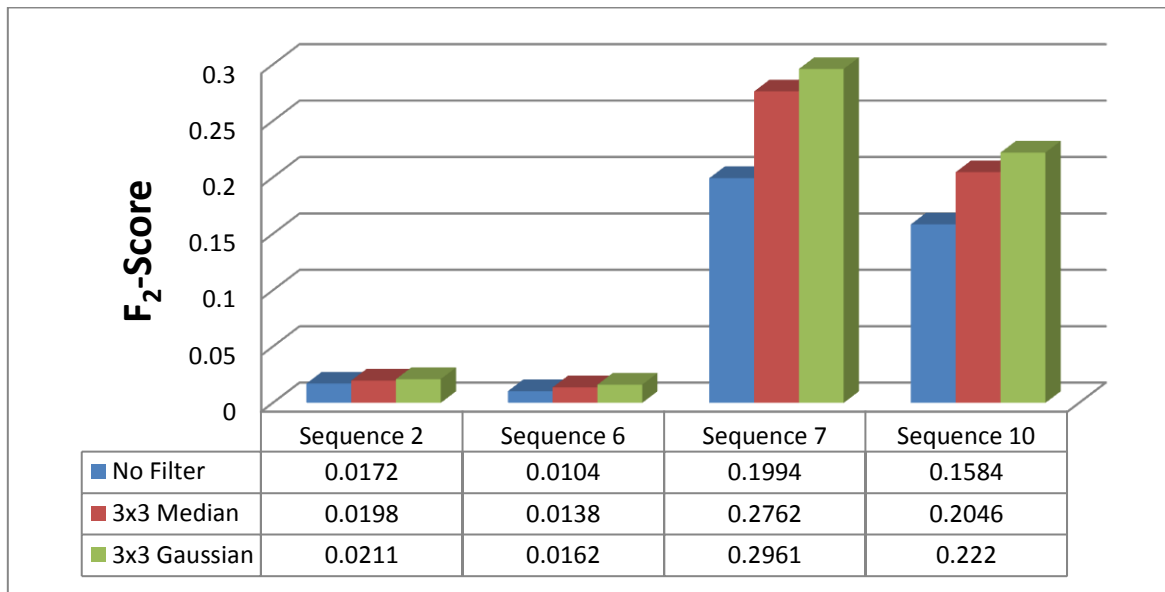


Figure 6-9 -  $F_2$ -Scores for various pre-filtering algorithms







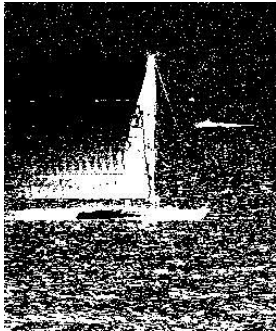
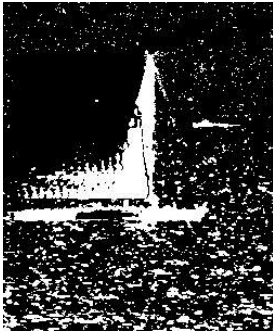



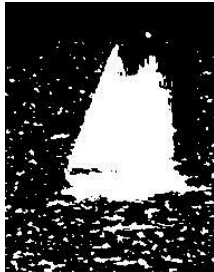
Table 6-4 shows the background subtraction results using each of these pre-filters on the test sequences. Attention is drawn to the outputs for the median and Gaussian pre-filters which show a removal of small erroneous pixels and a reduction in size of larger ones. One may also note that Gaussian pre-filtering is able to remove some of the smudging effects shown in sequence 10.

These smudging effects trailing the objects are simply artefacts of the threshold selection for the sequences. As the ship in sequence 10, for example, moves from right to left, its pixel values become part of the density estimate for pixels trailing it. This decreases the probability of seeing a sea pixel, and so when one is seen, this is may be marked as the foreground if the threshold is not set low enough.

It is clear that the use of a 3x3 Gaussian filter provides the best results and so this was used for testing in the next stages of the algorithm:

- Pre-filter choice: 3x3 Gaussian

Table 6-4 - Outputs for various pre-filters

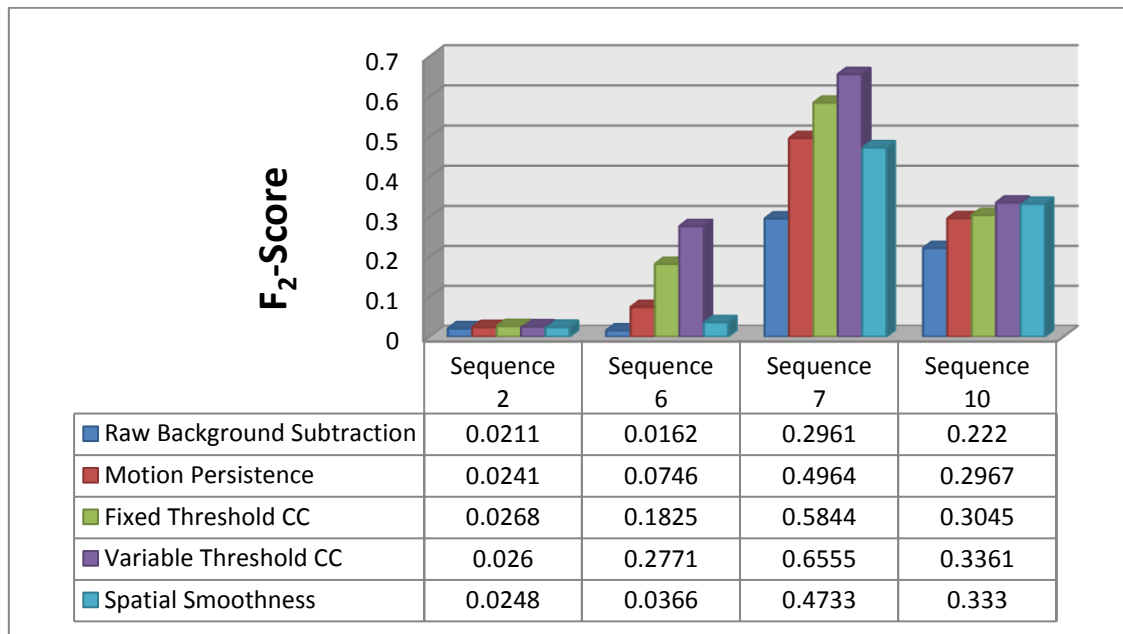
	No Filter	Median	Gaussian
Sequence 2			
Sequence 6			
Sequence 7			
Sequence 10			

### 6.5.2 Post-Filtering

Figure 6-10 shows the  $F_2$ -scores of various post-filtering methods. These have been applied to the output of the background subtraction used with a Gaussian pre-filter and so  $F_2$ -scores for the raw background subtracted output have been included for comparison. Every filtering method was able to improve the score for every sequence.  $F_2$ -scores for sequences with large objects are less sensitive to false positives as they do not compare much proportionally to the number of pixels in

the object. The converse is true for smaller objects such as sequence 6 where the largest improvement from filtering is shown.

As they are the most aggressive filtering methods, it comes as no surprise that the connected-component filters (both fixed and variable threshold) gave the biggest improvement in scores. One should bear in mind the possibility that if a target were too small, these filtering algorithms would remove it from the image and so there is an associated risk with using them. Due to its increasing threshold at the bottom of the image, the variable threshold connected-component algorithm was able to remove more false positives than the fixed threshold, producing the highest F-scores of all the filters. This algorithm yielded an average increase of 78% in  $F_2$ -scores for all test sequences.



**Figure 6-10 - Comparison of various  $F_2$ -scores for various filtering algorithms**

From Table 6-1 it is clear that detection results are highly dependent on the weather and size of the target object. If the object is too small (sequence 6), or the scene has too much glint (sequence 2) the results are poor. While post-filtering methods are able to remove some false positives in sequence 6, thereby increasing its score, glint results in large numbers of false positives in the image that are hard to remove (Figure 6-11).

Table 6-5 shows the actual outputs of the various filters for each of the sequences. These are again compared to the raw output of the background subtraction algorithm.

Visually it is easy to see in Table 6-5 why the connected-component filters have the highest scores, as there are very little false positives left in the image, while the closing operation was able to fill in a few false negative pixels in the objects. One is unable to differentiate between the output of the fixed and variable threshold filters as the majority of removed false positives are removed in the rest of the image.



Table 6-5 - Results of filtering for various sequences and filters



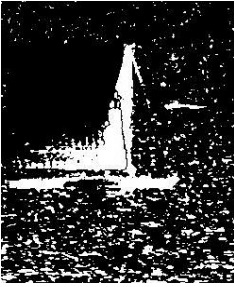
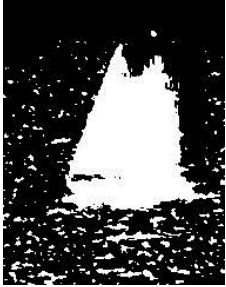


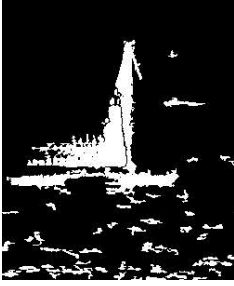
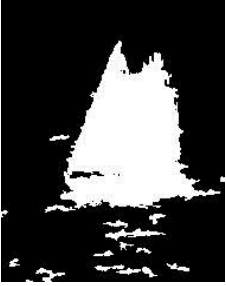


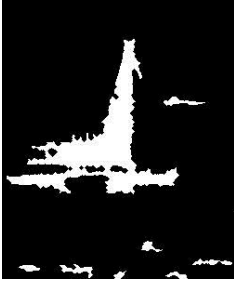



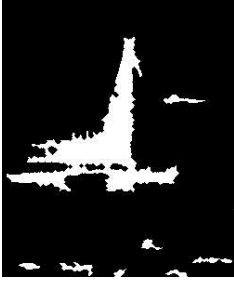



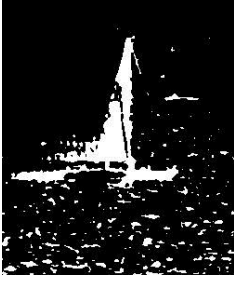
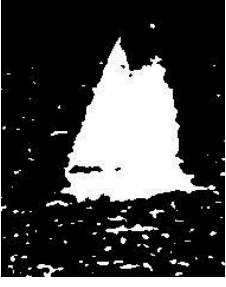
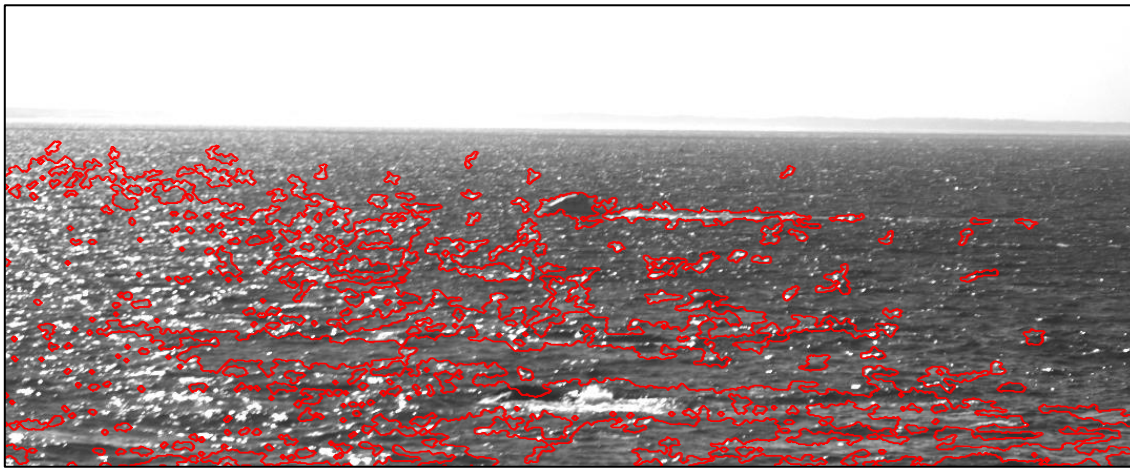
	Sequence 2	Sequence 6	Sequence 7	Sequence 10
Raw Background Subtraction				
Motion Persistence				
Fixed CC Size Threshold				
Variable CC Size Threshold				
Spatial Smoothness				

Figure 6-11 to Figure 6-14 show the output of the background subtraction superimposed on the entire image for each of the sequences. The large amount of glint and ocean movement illustrate why filtering methods are unable to improve  $F_2$ -scores in sequence 2 (Figure 6-11). Despite being considered false positives in the context of ship-tracking, the positives in the image do in fact correspond to actual motion in the sea; such as the wake from the ship.



**Figure 6-11 - Output from background subtraction overlaid on original image for sequence 2**



**Figure 6-12 - Output from background subtraction overlaid on original image for sequence 6**

The background subtraction algorithm's effectiveness is illustrated in sequence 7 (Figure 6-13) where the guide-line from the yacht (to the right of the top of the sail) is picked up by the background subtraction algorithm despite being almost invisible to the naked eye. Thanks to the variable threshold in this image, the majority of false positives were removed at the bottom of the

image, while still preserving the motion from the boat in the far off distance (to the right of the yacht).



**Figure 6-13 - Output from background subtraction overlaid on original image for sequence 7**

Simple inspection of the results for sequence 10 (Figure 6-14) shows that almost all the false positives were due to genuine motion from the sea in the image. The supposed false positive pixels at the bottom of the actual ship blob were again genuine motion due to its reflection in the water.



**Figure 6-14 - Output from background subtraction overlaid on original image for sequence 10**

### 6.5.3 Level Set Filtering

To test the level set segmentation technique, the output from the variable threshold connected-component algorithm was used as input data as it had the best results out of all the filtering methods. The energy functional used and details of its evolution are discussed in section 5.4.4.

Table 6-6 shows P-scores for each sequence and indicates which passed or failed at classification (recall, a pass is a P-score greater than 1 while a fail is a P-score less than or equal to 1). Apart from sequence 6, all the sequences were correctly classified with very good P-scores. A likely reason for sequence 6's failure is the similarity in shape of the blob around the ship with false positive blobs in the image (see Figure 6-12).

Table 6-6 - P-scores for various image sequences using Chan-Vese Energy as classification criteria

Sequence	P-Score	Classification Pass/Fail?
2	6.9316	Pass
6	0.4642	Fail
7	4.2569	Pass
10	13.1468	Pass

Table 6-7 shows actual segmentation results for the sequences that were correctly classified. While the green contour shows the ground truth for these objects, the red contour shows the resultant contour from segmentation. While sequence 2 can be considered a good segmentation, the smudging effects discussed above have caused poor segmentation results for sequences 7 and 10. The segmenting contours have tried to position themselves to include as many of these pixels as possible resulting in offsets from the ground truth.

**Table 6-7 - Segmentation results from image sequences that passed level set filtering.** *The boundary of the ground truth object is highlighted in green while the segmenting contour is highlighted in red*

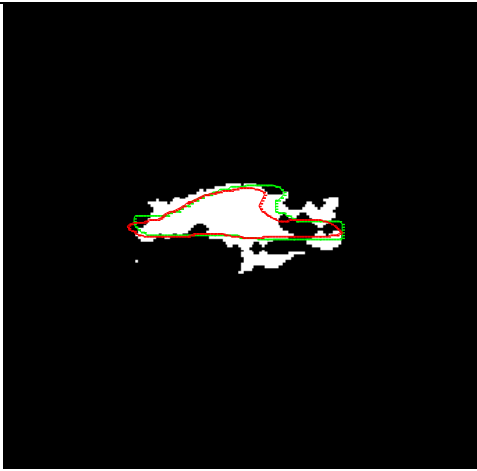
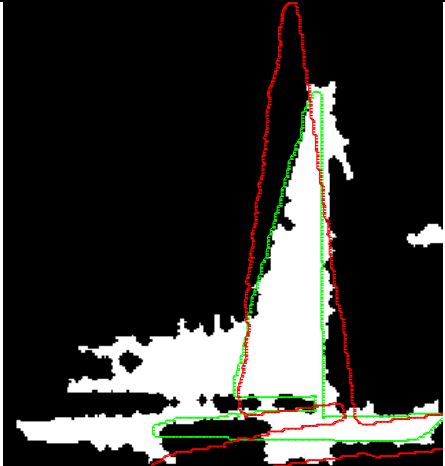
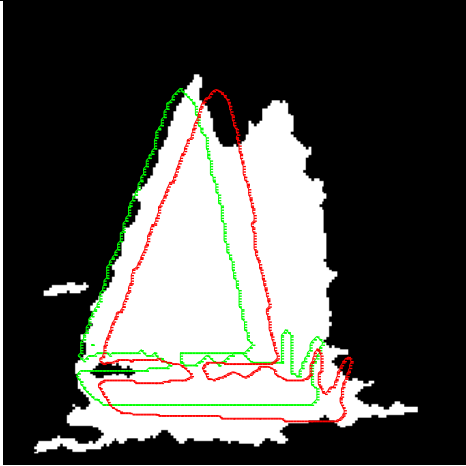
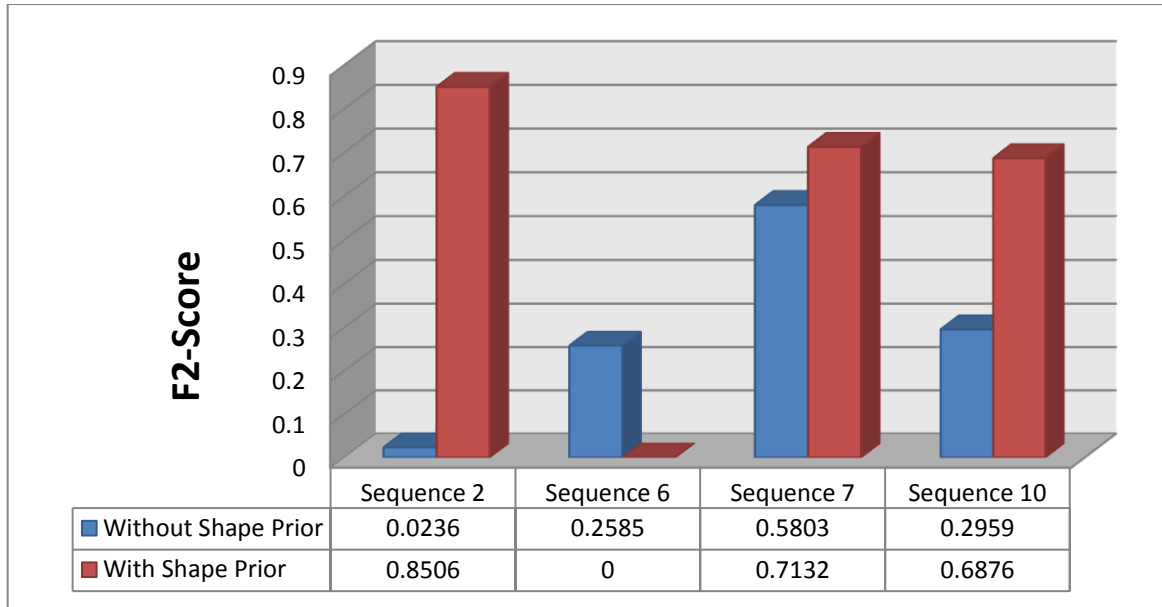
	Segmentation Result	Dice's Coefficient
Sequence 2		0.8698
Sequence 7		0.6113
Sequence 10		0.6667

Figure 6-15 shows  $F_2$ -scores for the output of the object detector using level set filtering compared to the output without it.



**Figure 6-15 -  $F_2$ -Scores for object detector with and without level set filtering with shape priors on its output**

Despite the incorrect classification for sequence 6, the level set filtering extensively improved  $F_2$ -scores for every other sequence. These scores may be compared with Szpak and Tapamo's system [8] whose highest  $F_2$ -scores were slightly less than 0.5. Sequence 2 was able to obtain the highest score despite having a large amount of glint in the image (Figure 6-11), a situation where Szpak and Tapamo's system failed.

## 6.6 Object Tracking Results

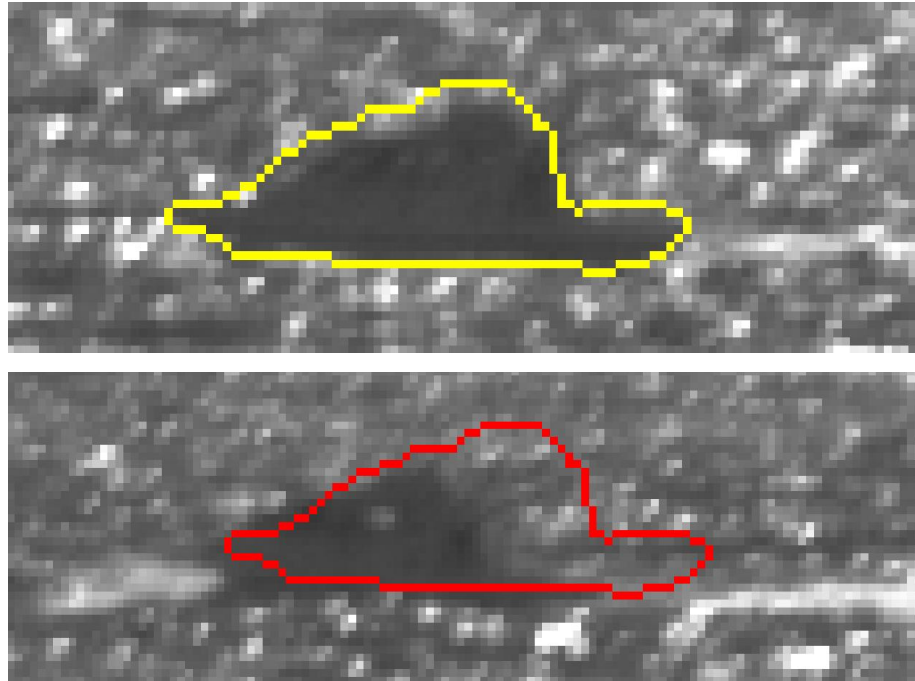
Given the parameters deduced in section 6.4.2, the object tracking performance using these parameters (namely, the FFT energy functional at 40 iterations per frame) can now be investigated. The tracking algorithm was run on the first 300 frames of each sequence using an initial contour derived from the ground truth object in the first frame as a target contour. This eliminates the dependency of the tracking algorithm on the object detection algorithm and allows its performance to be measured in isolation. Table 6-8 shows the tracker detection rate (TRDR), object tracking error (OTE) and contour tracking error (CTE) for each sequence.

**Table 6-8 – Performance metrics for each sequence.** *The sequences have been calculated using sequences of 300 frames*

Sequence	TRDR	OTE	CTE
<b>1</b>	1	1.8795	0.098
<b>2</b>	1	3.7144	0.225
<b>3</b>	1	1.4506	0.091
<b>4</b>	0.85	10.799	0.337
<b>5</b>	0.71	4.8505	0.206
<b>6</b>	0.38	7.2204	0.408
<b>7</b>	1	20.62	0.349
<b>8</b>	1	2.7759	0.331
<b>9</b>	0.7933	1.5088	0.075
<b>10</b>	1	2.6944	0.082

Detailed analysis of some of the sequences uncovers reasons for some of the above scores.

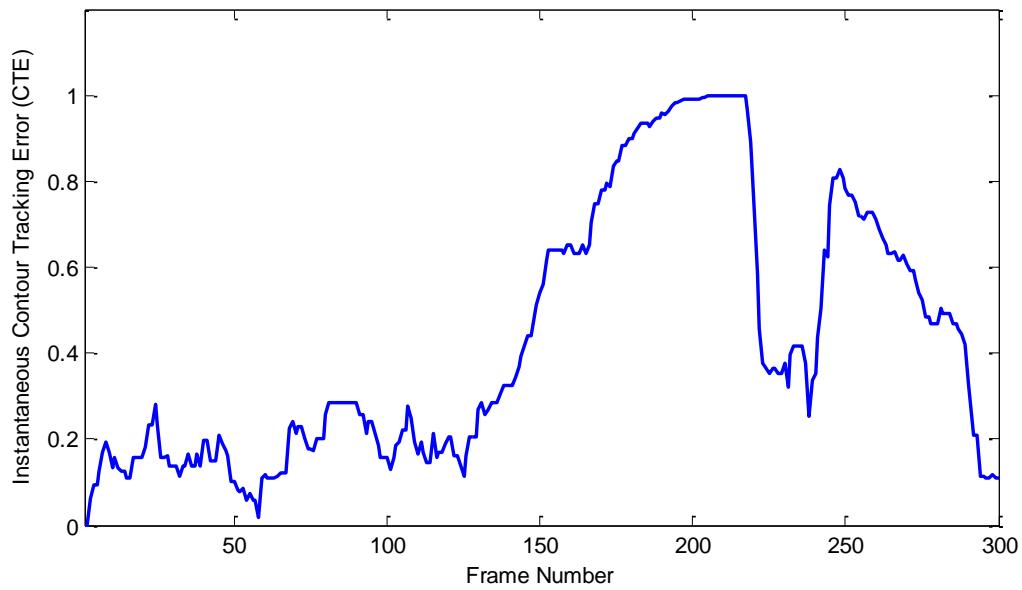
The reason for poor performance in sequence 4 can be attributed to change in frequency characteristics of the pixels of the tracked object as the sequence is run. Figure 6-16 shows the target contour in the first frame (above in yellow) and the tracking contour in the frame where it starts to drift away from the object. Attention is drawn to the bottom image in which glint from the sun's reflection in the object's window and a dark patch may have caused frequency components similar to the surrounding ocean, rendering the tracker unable to differentiate between the object and its surroundings.



**Figure 6-16 - Comparison of target contour from sequence 4 (yellow) and the frame in which the contour started to drift from the object (red)**

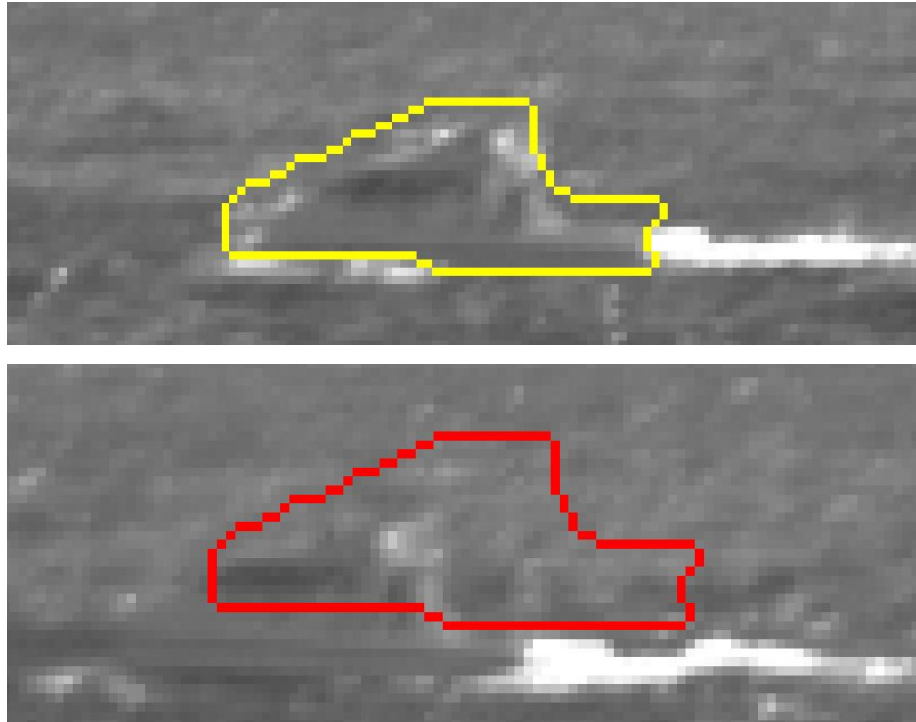
While the contour tracking error (CTE) is normally calculated as the average error over the frames that tracking was successful, Figure 6-17 shows the instantaneous CTE calculated for each particular frame regardless of success of tracking. This shows an increase in CTE as the contour starts to lose its object, however, it also shows a dip in error starting around frame 200. This was due to panning of the camera to the left (starting at frame 216) which was able to “save” the tracking by moving the object back into its contour. This explains the comparatively high tracking detection rate in Table 6-8.





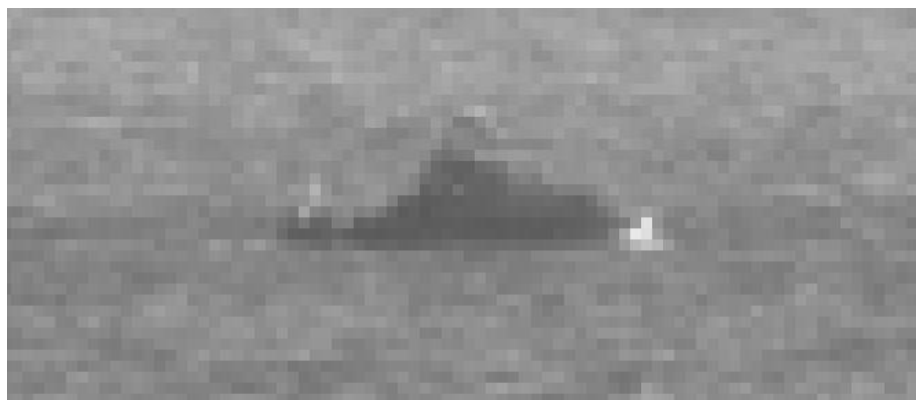
**Figure 6-17 - Instantaneous Contour Tracking Error vs. frame number for sequence 4**

For sequence 5, Figure 6-18 shows the target contour in the first frame (above in yellow) and the tracking contour in the frame where it starts to drift away from the object (bottom in red). Again, a change in object appearance can be attributed to error. In this case, glint from the sun strongly defines the borders of the object in the target frame, while the contour starts to drift when this glint is no longer present, making the target almost indistinguishable to the human eye.



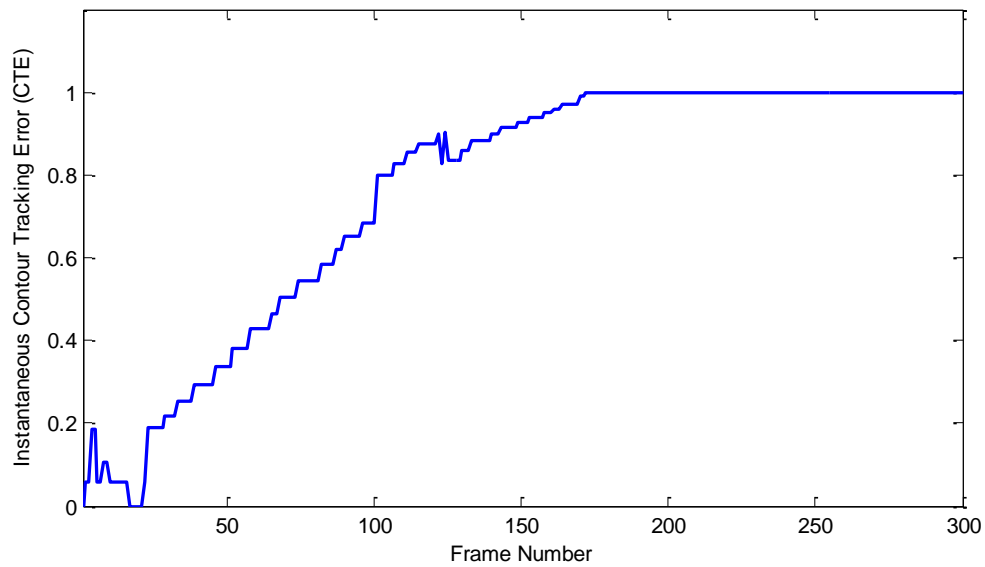
**Figure 6-18 - Comparison of target contour from sequence 4 (yellow) and the frame in which the contour started to drift from the object (red)**

Probable reasons for the error in sequence 6 include its small size and homogeneous appearance. Larger objects are able to incorporate more frequency information than their smaller counterparts, which in sequence 6's case has an object only about 80 pixels in size. This is made worse by the homogeneous nature of its pixels and the homogeneous nature of its surroundings as shown in Figure 6-19.



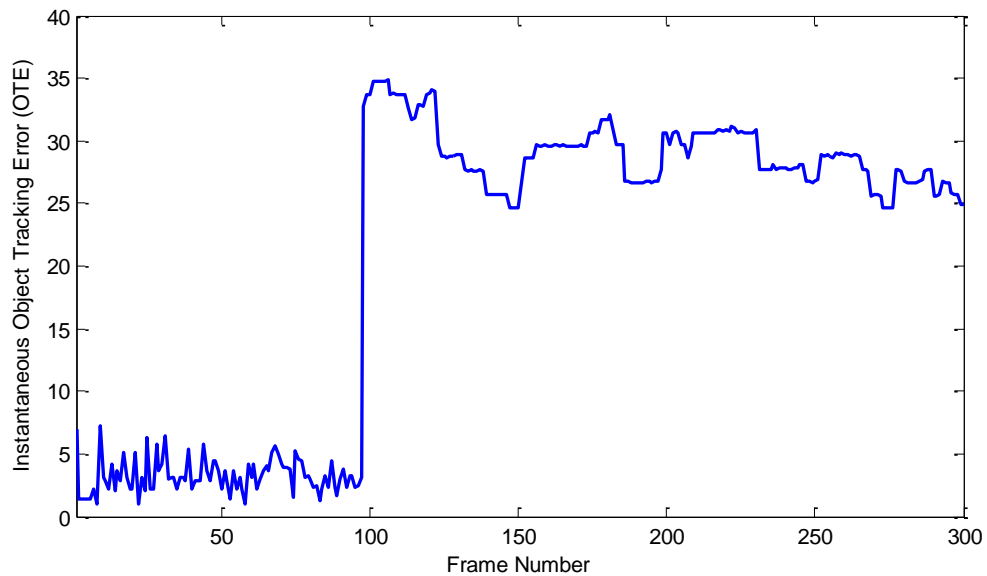
**Figure 6-19 - Homogeneous nature of pixels inside object for sequence 6 and surroundings**

This causes the tracking contour to start to drift almost immediately as tracking starts, as shown in the contour tracking error in Figure 6-20.



**Figure 6-20 - Instantaneous Contour Tracking Error vs. frame number for sequence 6**

Although the tracking contour successfully overlapped the object in every frame of sequence 7, the sequence has a comparatively high object tracking error of 20.62. A plot of the instantaneous object tracking error is shown in Figure 6-21. Around frame 100 the error jumps up and remains over 25 for the rest of the sequence.



**Figure 6-21 - Instantaneous Object Tracking Error vs. frame number for sequence 7**

Investigation of the output of the system (Figure 6-22) shows why this is the case: While frames proceeding 100 correctly track the object like the contour shown in frame 88, for all frames thereafter the contour is stuck below the object like that in frame 100. This gives the almost constant object tracking error seen in Figure 6-21. The most likely explanation for this anomaly is a local minimum in the energy functional below the object in which the level set contour gets stuck.



Frame 88

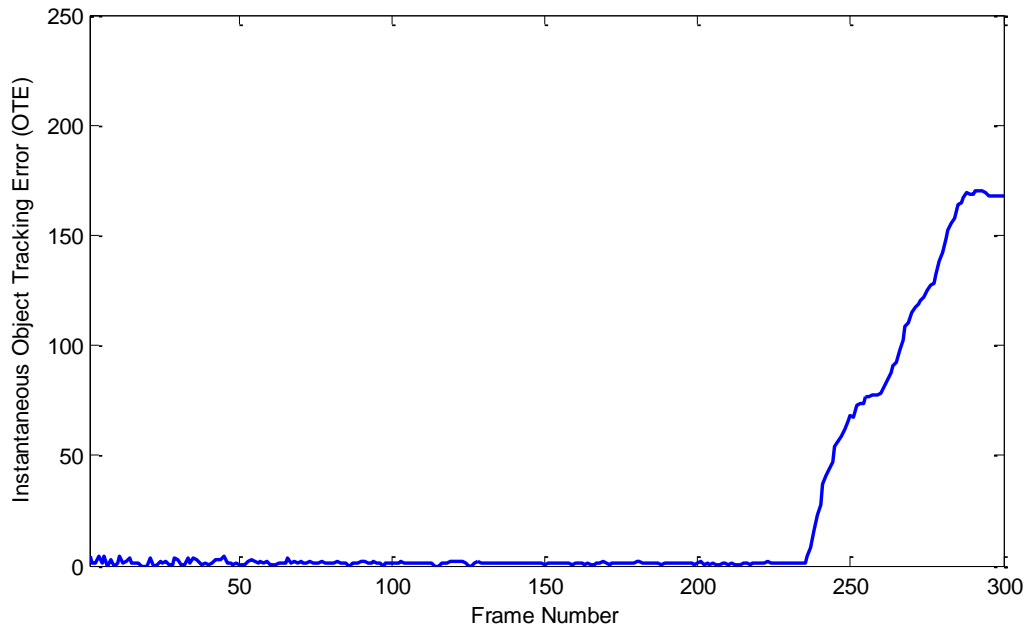


Frame 100

**Figure 6-22 - Comparison of system output at frame 88 and at frame 100**

The large contour error present in sequence 8 compared to other trackers can be explained by the object's small size. Even small variation of the contour around the object (which is evident from its small object tracking error) would result in large portions of the object to be incorrectly segmented.

Sequence 9 is an example of error produced by camera panning. At frame 235 the camera begins panning and the system contour is unable to keep up. For this reason the distance between the system contour's centroid and the object centroid continues to rise, thereby increasing the object tracking error. This doesn't necessary show in Table 6-8 as the ODE is calculated for frames where the two overlap, however a plot of instantaneous object tracking error in Figure 6-23 shows the gradual increase as panning occurs.



**Figure 6-23 -Instantaneous Object Tracking Error for sequence 9**

It is expected that if panning is slow enough and given more iterations per frame, the system would be able keep up with camera panning despite poor results presented here.

Unfortunately little comparison can be made with the object tracking performance of Szpak and Tapamo's work [8] as it only includes qualitative results for each sequence. It may be noted; however, that Szpak and Tapamo's system performs poorly in sequences with glint. The system presented is able to successfully track its target in every frame when tested on 2 of the 3 sequences containing glint.

## 6.7 Speed of Algorithm

Although speed of execution of a video tracking algorithm was discussed in the introductory chapter, it has not been mentioned since. In a realistic system, video tracking algorithms are required to run at about 30 frames per second. This gives a processing time of 33 milliseconds for each frame. The concept of execution speed of the video tracker developed has purposefully been ignored in order to focus on development of concepts and methodology. Since this work is focused on development of new algorithms it is appropriate to focus on these fundamental issues rather than address supplementary practical issues such as algorithm speed. These practical concerns may be optimised in future work for real-time computing.

## 6.8 Conclusion

This chapter presents and discusses the performance of the system. Details of the various performance metrics that have been devised to quantitatively measure performance of various parts of the system are presented. These include metrics to measure object detection, namely the F-score, the P-score and Dice's coefficient and metrics for object tracking, namely the tracking detection rates, object tracking error and contour tracking error. These are optimised for a very small data set and in no way can be considered completely optimal, nevertheless sample parameter values are found experimentally.

For the object detection: the background subtraction algorithm was found to have an optimal buffer size of 16 frames, frame spacing of 11 frames, and a threshold of 0.0117. The connected-component algorithms were found to have an optimal fixed threshold of 60 pixels and a variable threshold function of  $M(y) = 90y + 10$  respectively. The object tracking the frequency-based energy functional was shown to be optimal at 40 iterations per frame.

Finally the object detector and object tracker of the system are tested in isolation using these optimised parameters, and the results from each module are presented and discussed extensively. The pre-filters were initially tested and the Gaussian filter was shown to be the best across every sequence. Using the output from the background subtraction using a Gaussian filter, each of the post-background subtraction filters was tested. The best filtering algorithm was shown to be the variable threshold connected-component filtering algorithm, where  $F_2$ -scores were showed an average improvement of about 78% for all of the test sequences. The filtered sequences (using the variable threshold) were tested using the level set filtering algorithm. Of the 4 tested sequences, the filtering stage was about to successfully find correct objects in 3 of them. The  $F_2$ -scores of these sequences with level set filtering with shape priors can be compared to a system without this knowledge such as Szpak and Tapamo's [8]. Every sequence tested by this system where classification was successful showed a higher score than that obtained by Szpak and Tapamo's.

Segmentation accuracy for these sequences varied, and was largely influenced by the shape of the blob made by the object in the background subtracted frame. The object tracker was tested in isolation for all 10 of the sequences. Of these, 6 had 100% tracking detection rates. This system obtained 100% tracking detection rates in 2 of the 3 sequences containing glint: a situation in which Szpak and Tapamo's system performed poorly.

## Chapter 7 - Conclusion

This dissertation has investigated the use of prior knowledge to aid level set segmentation in video tracking for a maritime surveillance problem. Returning to the problem definition in Chapter 1, it can be stated that the research presented here successfully achieved solutions to the problems specified.

This chapter gives an overview of the dissertation, discussing important parts of each chapter. Finally it covers future areas of research that may be used to improve the system discussed and make it more robust for use in a real-life maritime surveillance scenario.

### 7.1 Summary of Dissertation

Chapter 1 includes a brief background of the automation of surveillance systems in maritime environments. A particular problem that these systems seek to solve is that of the ever-increasing amounts of piracy on today's waters. An increased awareness of a ship's surroundings is imperative to evade potential attacks. Compared to the use of multiple crew members on guard duty, automated surveillance systems do not grow tired and do not require the human resources that manual observation does. That being said, these video tracking systems are faced with a number of challenges, especially in a maritime environment, such as a moving background from a bobbing ship or confusion with moving waves for actual objects of interest. The motivation for this work is thus the use of prior knowledge of a ship's *shape* to differentiate between parts of the image that belong to waves and those that belong to actual objects of interest. The problem is defined in steps to be as follows: Investigation of various methods that allow the shape knowledge to be incorporated, a production of a model that uses one of these methods in a maritime environment, followed by testing of this video tracking system on real maritime data. Finally an outline of the dissertation is presented.

Chapter 2 initially explores the overall theory that is associated with the production of a video tracker. The choice of both object detection and object tracking is shown to form an integral part of the video tracker and various methods of achieving each step are reviewed. Object detectors can be grouped into those that detect single points of interest in the image, those that model the background to find moving objects in a video sequence, those that segment the image into regions of interest in expectation that objects form part of these regions and those that create intelligent agents that learn the views of an object and are able to find similar ones in an image. Object

trackers are highly dependent on the object detectors used: Point trackers are designed to match points detected in two consecutive video frames, kernel trackers are designed to track regions within simple geometric shapes while silhouette tracking (including level set segmentation) is used to track objects with complex outlines. A discussion on how these methods have been applied to video tracking in the maritime environment is presented. This includes the various basic methods presented in the past that attempt to characterise the ocean in some way and then remove it from the image in expectation that what remains are objects of interest. The various techniques to do this include characterising the ocean using frequency analysis, its grey-level histogram, and statistical features drawn from ocean tiles. While some methods have users indicate tiles belonging to the ocean prior to working, some surpass user intervention by transforming an image into points in feature-space and removing the largest cluster assuming that this belongs to ocean. This chapter concludes with a discussion on Szpak and Tapamo's implementation of level sets for tracking in [8] which forms a basis for the work presented in this dissertation.

As the object detector used in [8] is based upon a background modelling algorithm, a review of the various methods to achieve this are presented in full detail in Chapter 3. These can be grouped into techniques that produce a background model consisting of a single image and those that model pixels with a probability distribution. The single image model assumes foreground pixels are those that deviate from model-values while the probability model assumes they are those that are calculated to be unlikely.

As level set segmentation forms the basis for the object tracker in [8], Chapter 4 includes an in-depth analysis of the history and implementation of this method and how it has grown to be used today. A precursor to level set segmentation is active snakes. These methods express a curve in the image as a parametric spline and use forces to push it towards objects of interest. Level set segmentation solves a shortcoming of this method: its inability to handle changes in topography such as region splitting. Level set segmentation expresses a contour in the image as the zeroth level set of a higher dimensional function that is able to move around in 3D space allowing its contour in 2D to merge and split accordingly. Initially this function was evolved in an image to imitate the behaviour of active snakes. Variational methods, in contrast, define an energy function that is dependent on the level set function and is minimised if some pre-defined conditions are met. A differential equation is derived that evolves the level set function in a direction that minimises this energy thus moving the contour to desirable points in the image.



Since the focus of this dissertation is the incorporation of prior knowledge of an object's shape into the level set segmentation, Chapter 4 follows with a full review of the various methods used to achieve this. These can be grouped into methods that generate an additional term to a variational energy functional that penalises deviation from a known shape or methods that embed shape directly in the level set function by defining it as a function of some parameters. In addition, various works in literature present methods for incorporation of multiple shapes into the segmentation as well as methods to selectively use this knowledge in segmentation. Methods that incorporate multiple shapes create a shape model from a set of training shapes using a number of modelling methods while selective shape priors incorporate a labelling function that controls whether shape-knowledge is to be used. This labelling is dependent on how similar a contour is to a particular training shape and evolves in parallel with the level set function.

Chapter 5 includes explicit detail of the model used by the system. It moves from a high level overview of the system and then moves into finer aspects of both the object detector and object tracker, providing pseudo-code algorithms to clarify the details of the implementation. The object detector is a cascade of a pre-filter, background subtraction stage, a binary post-filtering stage and finally (if the user wishes) a level set shape prior filtering stage. The object tracker is composed of a model generation stage that receives its input from the object detector, and a tracking stage that achieves tracking by comparing objects in current frames with the model generated.

The level set segmentation formulation presented by Tsai *et al* [10] is the method used to incorporate prior knowledge of shape in the system and thus is discussed in detail. This is followed by a novel modification to this work that estimates energy gradient using a central difference scheme which allows the use of any arbitrary energy functional to be used with this formulation and avoid complex symbolic derivation.

The details of how this new model fits into both the object detector and object tracker are discussed. Image pre-filters may consist of a Gaussian filter or a Median filter. The background subtraction stage is of a probability modelling type and composed of a pixel-wise kernel density estimator, followed by a binary filtering stage that consists of one of a number of methods including filtering based on motion persistence, connected-component size filtering with a fixed and variable threshold and a spatial smoothness constraint proposed by Szpak and Tapamo [8]. The shape filtering stage uses a novel energy functional that is tailored to segment binary image outputs from the background modelling and filtering stages. The object tracker achieves tracking

using one of a number of novel energy functionals designed to compare candidate contours in current image frames with a target contour in the first frame of detection. These include functionals that compare a target and candidate contour using their histograms, frequency information, and statistical features.

Chapter 6 presents the results obtained from applying this system to a set of image sequences from a maritime environment. Various metrics used to measure the performance of the system and their meanings are covered. Object detection metrics include the  $F_2$ -score, used for measuring raw object detection accuracy; a novel P-score, used to measure effectiveness of shape filtering and Dice's coefficient, which measures level set segmentation accuracy. Object tracking metrics include the tracker detection rate, which measures the proportion of frames successfully tracked; object tracking error, which measures a rough estimation of how well these frames are tracked using a difference in centroids and a contour tracking error, which measures tracking error more accurately as average segmentation error for tracked frames.

Optimisation of the system and details of optimal parameters are discussed. These are optimised for a very small data set and in no way can be considered completely rigorous, however, the concepts used to determine them warrant presentation and could easily be used with a larger data set in future work. Parameters for the background subtraction and filtering stages were chosen to minimise segmentation error for a sample window around objects in sequences. For the background subtraction algorithm, these include a buffer size of 16 frames, frame spacing of 11 frames, and a threshold of 0.0117. The connected-component algorithms were found to have an optimal fixed threshold of 60 pixels and a variable threshold function of  $M(y) = 90y + 10$  respectively. In order to optimise the tracking algorithm the average Dice's coefficient for a single sequence was maximised with respect to tracking energy and iterations per frame. The frequency-based energy functional was shown to be optimal at 40 iterations per frame.

Finally the object detector and object tracker of the system are tested in isolation using these optimised parameters, and the results from each module are presented and discussed extensively.

Each of the pre-filters was tested in isolation and compared to the output of the optimised background subtraction algorithm without a pre-filter. Each pre-filter yielded an improvement in  $F_2$ -scores (at most about 0.9) for every test sequence. The Gaussian filter was shown to be the best pre-filter across every sequence.

Using the output from the background subtraction implemented with a Gaussian filter, each of the post-background subtraction filters was tested. Although a single test sequence only slightly increased its  $F_2$ -scores due to filtering (due to large false-positives in the image) the other test sequences showed drastic improvements in F-scores from all of the filtering algorithms. The best filtering algorithm was shown to be the variable threshold connected-component filtering algorithm, where  $F_2$ -scores were shown to have an average improvement of about 78% for all of the test sequences. The filtered sequences (using the variable threshold) were tested using the level set filtering algorithm. Of the 4 tested sequences, the filtering stage was able to successfully find correct objects in 3 of them. Segmentation accuracy for these sequences varied, and was largely influenced by the shape of the blob made by the object in the background subtracted frame.

The object tracker was tested in isolation for all 10 of the sequences. Of these, 6 had 100% tracking detection rates. Errors in tracking were attributed to the change in objects from their target models as the sequences progressed, camera panning, and the small size of objects in certain sequences.

The system was shown to perform better than one without prior shape knowledge such as that of Szpak and Tapamo [8] in both object detection and object tracking. In particular, the system continued to perform well even in situations where Szpak and Tapamo's system would fail.

## **7.2 Future Work**

While the system is able to successfully detect and track maritime objects, it still requires some assumptions that would not be true in a real-life situation. Future work on this system would allow for relaxation of these assumptions.

Firstly, the use of a background modelling algorithm to achieve object detection relies on a stationary camera to function. This is not the case in a real-life situation where a camera would most likely be mounted on the mast of a moving ship. An edge-based level set segmentation with shape priors for object detection rather than a filtered background subtraction algorithm could be used in the future. This would segment the raw image directly using a level set and would hopefully alleviate the requirement for a stationary background and allow cameras to be mounted on a bobbing ship. As background subtraction would not be used, the number of false positives from movement of waves would be reduced.

Secondly this system only uses a single shape prior that must be manually pre-set for every sequence that would not be feasible in a real-life system. To remove the reliance on the user a bank of multiple training shapes could be modelled using a method such as kernel density estimation in [50]. This model would then be used in replace of a fixed shape prior for segmentation.

It has already been noted that the system would be far more optimal given a larger training set. The concepts and methodology developed in this dissertation can be effortlessly implemented with a larger dataset and so future work could include further incorporation of training data into the various subsystems discussed to produce more optimal parameters.

Finally, the system runs on a single-threaded MATLAB implementation and is thus quite slow. Future work can entail optimising this algorithm for parallel processing, and implementing it in a faster framework such as C++.

## References

- [1] G Stubblefield and B Parlato, "Condition Yellow," *Passagemaker*, p. 85, Winter 1999.
- [2] ICC Commercial Crime Services. (2012, January) ICC Commercial Crime Services. [Online].  
<http://www.icc-ccs.org/news/711-piracy-attacks-in-east-and-west-africa-dominate-world-report>
- [3] M Bruyneel. (2001, February) Modern Day Piracy Statistics. [Online].  
<http://home.wanadoo.nl/m.bruyneel/archive/modern/figures.htm>
- [4] BBC Radio World Service. (2008, March) BBC. [Online].  
[http://www.bbc.co.uk/worldservice/documentaries/2008/03/080303\\_pirates\\_prog2.shtml](http://www.bbc.co.uk/worldservice/documentaries/2008/03/080303_pirates_prog2.shtml)
- [5] JG Sanderson, MK Teal, and TJ Ellis, "Characterisation of a Complex Maritime Scene Using Fourier Space Analysis to Identify Small Craft," in *Proceedings 7th International Conference on Image Processing and Its Applications*, Manchester, 1999, pp. 803-807.
- [6] AAW Smith and MK Teal, "Identification and Tracking of Maritime Objects in Near-Infrared Image Sequences for Collision Avoidance," in *Proceedings IEEE 7th International Conference on Image Processing and its Applications*, Manchester, 1999, pp. 250-254.
- [7] P Voles, M Teal, and J Sanderson, "Target Identification in a Complex Maritime Scene," in *IEEE Colloquium on Motion Analysis and Tracking*, London, 1999, pp. 15/1-15/4.
- [8] ZL Szpak and JR Tapamo, "Maritime Surveillance: Tracking Ships Inside a Dynamic Background using a Fast Level-set," *Expert Systems with Applications: An International Journal*, Vol. 38, No. 6, pp. 6669-6680, 2011.
- [9] D Socek, D Culibrk, O Marques, H Kalva, and B Furht, "A Hybrid Color-Based Foreground Object Detection Method for Automated Marine Surveillance," in *ACIVS'05 Proceedings of the 7th international conference on Advanced Concepts for Intelligent Vision Systems*, Antwerp, 2005, pp. 340-347.
- [10] A Tsai, A Yezzi, W Wells, C Tempany, D Tucker, A Fan, W Eric Grimson, A Willsky ., "A Shape-Based Approach to the Segmentation of Medical Imagery Using Level Sets," *IEEE*

- Transactions on Medical Imaging*, Vol. 22, No. 2, vol. 22, pp. 137-154, 2003.
- [11] E Maggio and A Cavallaro, *Video Tracking: Theory and Practise*. New Delhi: Wiley, 2011.
- [12] A Yilmaz, O Javed, and M Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, Vol. 38, No. 4, pp. 13-20, 2006.
- [13] Microsoft. (2012, May) Kinect for Xbox 360. [Online]. [www.xbox.com/kinect/](http://www.xbox.com/kinect/)
- [14] H Moravec. (1980) Stanford University. [Online]. <http://www.frc.ri.cmu.edu/~hpm/project.archive/robot.papers/1975.cart/1980.html.thesis/index.html>
- [15] C Harris and M Stephens, "A Combined Corner and Edge Detector," in *Proceedings 4th Alvey Vision Conference*, Cambridge, 1988, pp. 147-151.
- [16] DG Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.
- [17] R Gonzalez and R Woods, *Digital Image Processing*. New Jersey: Prentice Hall, 2002.
- [18] D. Comaniciu and P. Meer, "Mean Shift Analysis and Applications," in *Proceedings IEEE International Conference on Computer Vision*, Kerkyra, 1999, p. 1197.
- [19] J Shi and J Malik, "Normalized Cuts and Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 888-905, 2000.
- [20] Z Wu and R Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and its Applications to Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 11, no. 11, pp. 1101-1113, 1993.
- [21] P Viola, MJ Jones, and D Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, Nice, 2003, p. 734.

- [22] V Salari and IK Sethi, "Feature Point Correspondence in the Presence of Occlusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 1, pp. 87-91, 1990.
- [23] TJ Broida and R Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 1, pp. 90-99, 1986.
- [24] D Comaniciu, V Ramesh, and P. Meer, "Kernel-Based Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, pp. 564-575, 2003.
- [25] D Huttenlocher, JJ Noh, and WJ Rucklidge, "Tracking Non-Rigid Objects in Complex Scenes," in *Proceedings 4th International Conference Computer Vision*, Berlin, 1993, pp. 93-101.
- [26] M Bertalmio, G Sapiro, and G Randall, "Morphing Active Contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 7, pp. 733-737, 2000.
- [27] P Voles and M Teal. (1998) Maritime Scene Segmentation. [Online]. [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/VOLES/marine.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VOLES/marine.html)
- [28] S Cheung and C Kamath, "Robust Techniques for Background Subtraction in Urban Traffic Video," in *Proceedings Video Communications and Image Processing*, San Jose, January 2004, pp. 881-892.
- [29] R Cutler and L Davis, "View-based Detection and Analysis of Periodic Motion," *Proceedings 14th International Conference on Pattern Recognition*, vol. 1, pp. 495-500, 1998.
- [30] N McFarlane and C Schofield, "Segmentation and Tracking of Piglets in Images," *Machine Vision and Applications*, Vol. 1, No. 1, pp. 187-193, 1995.
- [31] C Stauffer and W. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Ft. Collins, 1999, pp. 246-252.
- [32] A Elgammal and R Duraiswami, "Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance," *Proceedings of the IEEE*,

Vol. 90, No. 1, pp. 1151-1163, July 2002.

- [33] R Gutierrez-Osuna. (2012, February) PRISM @ TAMU. [Online].  
[http://research.cs.tamu.edu/prism/lectures/pr/pr\\_l7.pdf](http://research.cs.tamu.edu/prism/lectures/pr/pr_l7.pdf)
- [34] BW Silverman, "Density Estimation for Statistics and Data Analysis," *Monographs on Statistics and Applied Probability*, Vol. 1, No. 1, pp. 1-22, 1986.
- [35] A Maistrou. (2008) Level Set Methods - Overview. [Online].  
<http://campar.in.tum.de/twiki/pub/Chair/TeachingSs08EvolvingContoursHauptseminar/ActiveContours.pdf>
- [36] M Kass, A Witkin, and D Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321-331, 1988.
- [37] T Chan and LA Vese, "Active Contours Without Edges," *IEEE Transactions on Image Processing*, Vol. 10, No. 2, pp. 266-277, 2001.
- [38] S Osher and J Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, Vol. 79, No. 1, pp. 12-49, 1987.
- [39] N Paragios and R Deriche, "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects," *IEEE Transactions Pattern Analysis and Machine Intelligence*, Vol. 22, No. 3, pp. 266-280, 2000.
- [40] D Cremers, "Dynamical Statistical Shape Priors for Level Set Based Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 8, pp. 1262-1273, 2006.
- [41] MA Grayson, "The Heat Equation Shrinks Embedded Plane Curves to Round Points," *Journal of Differential Geometry*, Vol. 26, No. 2, pp. 285-314, 1987.
- [42] S Osher and R Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. New York: Springer, 2003.



- [43] V Caselles, F Catté, T Coll, and F Dibos, "A Geometric Model for Active Contours in Image Processing," *Numerische Mathematik, Vol. 66, No. 1*, vol. 66, pp. 1-31, 1993.
- [44] D Cremers, N Sochen, and C Schnorr, "Towards Recognition-Based Variational Segmentation Using Shape Priors and Dynamic Labeling," in *Scale Space'03 Proceedings of the 4th international conference on Scale space methods in computer vision*, Isle of Skye, 2003, pp. 388-400.
- [45] T Chan and W Zhu, "Level Set Based Shape Prior Segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 2005, pp. 1164-1170.
- [46] M Rousson and N Paragios, "Shape Priors for Level Set Representations," in *ECCV '02 Proceedings of the 7th European Conference on Computer Vision-Part II*, Copenhagen, 2002, pp. 78-92.
- [47] D Cremers, N Sochen, and C Schnorr, "A Multiphase Dynamic Labeling Model for Variational Recognition-Driven Image Segmentation," *International Journal of Computer Vision, Vol. 66, No. 1*, pp. 67-81, 2005.
- [48] K Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine, Vol. 2, No. 6*, pp. 559-572, 1901.
- [49] T Jehan, "Creating Music by Listening PHD Thesis," 2005. [Online]. [web.media.mit.edu/~tristan/Papers/PhD\\_Tristan.pdf](http://web.media.mit.edu/~tristan/Papers/PhD_Tristan.pdf)
- [50] D Cremers, S Osher, and S Soatto, "Kernel Density Estimation and Intrinsic Alignment for Shape Priors in Level Set Segmentation," *International Journal of Computer Vision, Vol. 60, No. 3*, pp. 335-351, 2006.
- [51] D Richtmeyer and K.W Morton, *Difference Methods for Initial Value Problems*, 2nd ed. New York: Wiley, 1967.
- [52] A Harvey and V Oryshchenko, "Kernel Density Estimation for Time Series Data,"

*International Journal of Forecasting*, Vol. 28, No. 1, pp. 3-14, January 2012.

- [53] A Yezzi and A Tsai, "A Statistical Approach to Snakes for Bimodal and Trimodal Imagery," in *Proceedings of the International Conference on Computer Vision*, Kerkyra, 1999, p. 898.
- [54] G Piella and H Heijmans, "A New Quality Metric for Image Fusion," in *Proceedings IEEE Conference on Image Processing*, Barcelona, 2004, pp. 173–176.
- [55] G Hripcsak and A Rothschild, "Agreement, the F-Measure, and Reliability in Information Retrieval," *Journal of the American Medical Informatics Association*, Vol. 12, No. 3, pp. 296-298, May-Jun 2005.
- [56] M Krishnaveni and V Radha, "Quantitative evaluation of Segmentation algorithms based on level set method for ISL datasets," *International Journal on Computer Science and Engineering*, Vol. 3, No. 2, pp. 2361-2369, June 2011.
- [57] LR Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, Vol. 26, No. 3, pp. 297-302, 1945.
- [58] F Bashir and F Porikli, "Performance Evaluation of Object Detection and Tracking Systems," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, New York, 2006, pp. 7-14.
- [59] K Dobbin and R Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC Medical Genomics*, Vol. 41, No. 7, pp. 1-31, April 2011.

## Bibliography

- [60] ME Leventon, WEL Grimson, and O Faugeras, "Statistical Shape Influence in Geodesic Active Contours," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Hilton Head, 2000, pp. 316 - 323.
- [61] T Bailloeul, V Prinet, B Serra, and P Marthon, "Spatio-temporal Prior Shape Constraint for Level Set Segmentation," in *EMMCVPR'05 Proceedings of the 5th international conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, St. Augustine, 2005, pp. 503-519.
- [62] Y Shi and W Karl, "Real-time Tracking Using Level Sets," in *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 2005, pp. 34-41.
- [63] P Viola and M Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, 2001, pp. 137-154.
- [64] D Mumford and J Shah, "Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems," *Communications on Pure and Applied Mathematics*, Vol. 42, No. 5, pp. 577–685, 1989.
- [65] T Riklin-Raviv, N Sochen, and N Kiryati, "Mutual Segmentation with Level Sets," in *Proceedings IEEE CVPR Workshop on Perceptual Organisation in Computer Vision*, New York, 2006, pp. 169–176.
- [66] A Elyasi, Y Ganjdanesh, K Kangarloo, Mi Hossini, and M Esfandyari, "Level Set Segmentation Method in Cancer's Cells Images," *Journal of American Science*, Vol. 7, No. 2, pp. 196-204, 2011.
- [67] C Li, C Xu, C Gui, and MD Fox, "Level Set Evolution Without Re-initialization: A New Variational Formulation," in *Proceedings Computer Vision and Pattern Recognition*, San Diego, 2005, pp. 162-167.

- [68] C Stauffer and W Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 747-757, 2000.
- [69] R Valentine. (2007) Image Segmentation with the Mumford Shah Functional. [Online]. <http://coldstonelabs.org/files/science/math/Intro-MS-Valentine.pdf>
- [70] TN Tan, GD Sullivan, and KD Baker, "Fast Vehicle Localisation and Recognition Without Line Extraction and Matching," in *Proceedings of The 5th British Machine Vision Association and Society for Pattern Recognition*, York, 1994, pp. 85-94.
- [71] C Li and C Xu, "Distance Regularized Level Set Evolution and Its Application to Image Segmentation," *IEEE Transactions on Image Processing*, Vol. 19, No. 12, pp. 3243-3254, 2010.
- [72] A Elgammal and R Duraiswami, "Background and Foreground Modelling Using Nonparametric Kernel Density Estimation for Visual Surveillance," *Proceedings of the IEEE*, Vol. 90, No. 7, pp. 1151-1163, July 2002.
- [73] J Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, pp. 308-313, 1965.
- [74] M Rousson and D Cremers, "Efficient Kernel Density Estimation of Shape and Intensity Priors for Level Set Segmentation," in *Proceedings International Conference on Medical Image Computing and Computer-Assisted Intervention*, Palm Springs, 2005, pp. 757-764.
- [75] W Yasnoff, J Mui, and J Bacus, "Error Measures for Scene Segmentation," *Pattern Recognition*, Vol. 9, No. 1, pp. 217-231, May 2003.
- [76] M Rousson, N Paragios, and R. Deriche, "Implicit Active Shape Models for 3d Segmentation in MRI Imaging," in *Proceedings Medical Image Computing and Computer Assisted Intervention*, St. Malo, 2004, pp. 209-216.
- [77] N Paragios, M Rousson, and V Ramesh, "Matching Distance Functions: A Shape-to-Area Variational Approach for Global-to-Local Registration," in *EECV*, Berlin, 2002, p. 775-789.
- [78] C Bibby and I Reid, "Real-time Tracking of Multiple Occluding Objects using Level Sets," in

*Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, 2010, pp. 1307-1314.

- [79] V Prisacariu and I Reid, "Nonlinear Shape Manifolds as Shape Priors in Level Set Segmentation," in *Proceedings 2011 Conference on Computer Vision and Pattern Recognition*, Colorado Springs, 2011, pp. 2185 - 2192.