Aerial Detection of Ground Moving Objects

by

©Agwad Hammad ElSayed ElTantawy

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

2018

St. John's Newfoundland

Abstract

Automatic detection of ground moving objects (GMOs) from aerial camera platforms (ACPs) is essential in many video processing applications, both civilian and military. However, the extremely small size of GMOs and the continuous shaky motion of ACPs present challenges in detecting GMOs for traditional methods. In particular, existing detection methods fail to balance high detection accuracy and real-time performance.

This thesis investigates the problem of GMOs detection from ACPs and overcoming the challenges and drawbacks that exist in traditional detection methods. The underlying assumption used in this thesis is based on principal component pursuits (PCP) in which the background of an aerial video is modelled as a low-rank matrix and the moving objects are modelled as sparse corrupting this video. The research in this thesis investigates the proposed problem in three directions: (1) handling the shaky motion in ACPs robustly with minimal computational cost, (2) improving the detection accuracy and radically lowering false detections via penalization term, and (3) extending PCP's formulation to achieve adequate real-time performance.

In this thesis, a series of novel algorithms are proposed to show the evolution of our research towards the development of KR-LNSP, a novel robust detection method which is characterized by high detection accuracy, low computational cost, adaptability to shaky motion in ACPs, and adequate real-time performance. Each of the proposed algorithms is intensively evaluated using different challenging datasets and

compared with current state-of-the-art methods.

DEDICATION

This thesis is dedicated to my daughter (NADA) and my father who left our world in the middle of my Ph.D journey (GOD HAS MERCY ON MY FATHER AND MAKES HIS PLACE HEAVEN).

Acknowledgments

First thank God. Then I am immensely grateful to my supervisor, Dr. Mohamed S. Shehata, for providing a very good research environment, his absolute help, guidance, and encouragement during the period of my research. Also, I would like to offer my sincere thanks to my family that is always by my side during my life journey.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

B.P.	Best Performance
S.P.	Second Best Performance
ACPs	Aerial Camera Platforms
GMOs	Ground Moving Objects
PCP	Principal Component Pursuit
PCA	Principal Component Analysis
S-PCP	Sequential-based Principal Component Pursuit
B-PCP	Batch-based Principal Component Pursuit
IALM	Inexact Augmented Lagrange Multiplier
APG	Accelerated Proximal Gradient
I-MARO	MAtrix Rank Optimization method with Inexact Newton method
N-MARO	MAtrix Rank Optimization with weighted Newton method
Spring-MARO	Spring model with MAtrix Rank Optimization method
UT-MARO	Unscented Transformation with MAtrix Rank Optimization method
KR-MARO	Kinematic Regularization and MAtrix Rank Optimization method
LNSP	Local Null Space Pursuit method
KR-LNSP	Kinematic Regularization with Local Null Space Pursuit method
RASL-APG	Robust Alignment by Sparse and Low-rank decomposition method that uses APG
RASL-IALM	Robust Alignment by Sparse and Low-rank decomposition method that uses IALM
DECOLOR	DEtecting Contiguous Outliers in the LOw-rank Representation method
3TD	3 Term Decomposition method
COROLA	Contiguous Outliers Representation via Online Low-rank Approximation method
GROUSE	Grassmanian Rank-One Update Subspace Estimation algorithm
GRASTA	Grassmannian Robust Adaptive Subspace Tracking Algorithm,
ROSETA	Robust Online Subspace Estimation and Tracking Algorithm
GOSUS	Grassmannian Online Subspace Updates with Structured-sparsity
amFastPCP	Fast Principal Component Pursuit method
IncPCP	Incremental Principal Component Pursuit method
ReProCS	Recursive Projected Compressive Sensing method
PracReProCSpPCA	Practical recursive projected Compressive Sensing method

List of Symbols

Operations and functions:

$\boldsymbol{vec(.)}$	An operation of converting a matrix to a column vector: $\mathbb{R}^{x \times y} \longrightarrow \mathbb{R}^{z}$
$\boldsymbol{vec^{\dagger}(.)}$	An operation of converting a column vector to a matrix: $\mathbb{R}^{z} \longrightarrow \mathbb{R}^{x \times y}$
$\boldsymbol{\|.\|_0}$	Zero norm ($\ell_0 - norm$)
$\boldsymbol{\|.\|_1}$	First norm ($\ell_1 - norm$)
$\boldsymbol{\|.\|_*}$	Nuclear norm ($\ell_* - norm$)
$\boldsymbol{\|.\|_f^2}$	Frobenius norm
$\boldsymbol{\langle x, y \rangle}$	Inner product
$\boldsymbol{\mathcal{L}(.)}$	Lagrange function
$\boldsymbol{Rank(.)}$	Matrix rank
$\boldsymbol{\mathcal{S}_{\alpha}(x)}$	Soft thresholding, $\mathcal{S}\alpha(x) = sign(x) \times max\{\|x\| - \alpha, 0\}$
$\boldsymbol{\overline{x}}$	Sub-vector of vector x
$\bullet$	Alignment Operator

Video Frame Related Symbols:

$\boldsymbol{w}$	A video frame width
$\boldsymbol{h}$	A video frame height
$\boldsymbol{n}$	A video frame resolution $n = w \times h$
$\boldsymbol{m}$	number of frames
$\boldsymbol{f}$	A video frame, $f \in \mathbb{R}^{w \times h}$
$\tilde{\boldsymbol{f}}$	A video frame after frame alignment operation using a transformation matrix
$\boldsymbol{F}$	A matrix of frames where each of its columns is a video frame $vec(f)$, $F \in \mathbb{R}^{m \times n}$
$\tilde{\boldsymbol{F}}$	Frames matrix with aligned frames
$\boldsymbol{set\_F}$	subset of frames matrix F
$\boldsymbol{b}$	Background in a video frame, $b \in \mathbb{R}^{w \times h}$
$\boldsymbol{B}$	A matrix of backgrounds, each of its columns is vector $vec(b)$, $B \in \mathbb{R}^{m \times n}$
$\boldsymbol{o}$	Moving objects in a video frame, $o \in \mathbb{R}^{w \times h}$
$\boldsymbol{O}$	A matrix of moving object, each of its columns is a vector $vec(o)$, $O \in \mathbb{R}^{m \times n}$
$\check{\boldsymbol{O}}$	A matrix of Candidate moving objects, $\check{O} \in \mathbb{R}^{m \times n}$
$\boldsymbol{CMO}$	Collection of pixels that form a candidate moving object
$\boldsymbol{d}$	False detection in a video frame, $d \in \mathbb{R}^{w \times h}$
$\boldsymbol{D}$	A matrix of false detections, each of its columns is a vector $vec(d)$, $D \in \mathbb{R}^{m \times n}$

Spring Related Symbols:

$\oint$	Combination of compression and torsion springs
$\boldsymbol{SM}$	Spring Model
$\boldsymbol{SF}$	The force of spring $\oint$
$\boldsymbol{CSF}$	Compression spring force
$\boldsymbol{TSF}$	Torsion spring force
$\boldsymbol{TF}$	Total Spring Forces
$\boldsymbol{\Gamma}$	Kinematic regularization term
$\boldsymbol{LM}$	A landmark which is a window in b

Other General Symbols:

$\boldsymbol{i}, \boldsymbol{j}$, $\boldsymbol{Omega}$, and $\boldsymbol{k}$	indexes
$\boldsymbol{\mu}, \boldsymbol{\kappa}$, $\boldsymbol{\xi}$, $\boldsymbol{lambda}$, $\boldsymbol{zeta}$, and $\boldsymbol{\epsilon}$	scalar values
$\boldsymbol{Y}$	Lagrange multiplier
$\boldsymbol{T}$	Transformation domain
$\triangle \boldsymbol{t}$	Newton step
$\boldsymbol{\mathcal{N}}$	Null space
$\boldsymbol{\mathcal{N}}^{\dagger}$	pseudo inverse of null space
$\boldsymbol{J}$	Jacobian
$\boldsymbol{J}^{\dagger}$	pseudo inverse of Jacobian
$\boldsymbol{\eta}$	Forcing term
$\boldsymbol{\theta}_x$	Angle between the locations of x in two frames
$\triangle \boldsymbol{x}$	Differences in X-axis
$\triangle \boldsymbol{y}$	Differences in Y-axis
$\boldsymbol{\Sigma}$	covariance matrix
$\boldsymbol{Mtch}_i$	SURF feature points in f_i
$\boldsymbol{dim}$	the dimensions SURF feature points

Chapter 1

Introduction

1.1 Overview

The past few decades have witnessed a massive revolution in computer vision concepts, theorems, and algorithms which in turn have had a significant impact in our modern life. Using computer vision in different applications saves human life, time and money, and in some cases computer vision offers more precise results than human visual perception. One of the computer vision processes that occupies a great importance in civilian and military applications is detecting ground moving objects (GMOs). For example, GMOs detection allows an automatic reporting of traffic conditions in a specific section of a road. Also, GMOs detection can be used to help secure country's border by warning security forces when an object to security interest is approaching. This critical role of GMOs detection, applied to a growing number of domains, is motivating researchers to investigate novel methods to achieve cost-effective, robust and computationally efficient detection. Therefore, the primary aim driving research in GMOs detection is to achieve a mature technological design for indoor environments and stationary camera applications.

With the use of aerial camera platforms (ACPs), such as unmanned air vehicles (UAVs), drones, balloons, etc., computer vision applications (e.g. [1], [2]) become more effective. This is due to the bird-eye view of ACP that allows a comprehensive analysis of ground scenes. However, GMOs detection becomes more challenging because of the shakiness of the ACPs and the relatively small size of GMOs. Therefore, a lot of research effort has been directed toward resolving these challenges. Nevertheless, GMOs detection methods still suffer from a lack of accuracy and high computational cost.

With this in mind, this thesis investigates novel methods to resolve the problems associated with detecting GMOs for ACPs. The backbone of this study is based on the concept of principal component pursuits (PCP) where the background and the moving objects are modelled as a low-rank and sparse matrices, respectively. Starting from the PCP concept, we handle the camera motion of ACPs robustly, improve the sparse modelling of the moving objects to enhance the accurate detection, and generalize the low-rank matrix modelling of the background to achieve real-time performance.

The rest of this chapter is organized as follows. The motivation which stands behind this thesis will be discussed in the next section. Next, the objective of this thesis will be stated, followed by an explanation of the thesis contributions.. Finally, the thesis outline is sketched.

1.2 Research Motivation

A growing trend in the application of computer vision relates to the capture and analysis of aerial imagery from ACPs. However, using ACPs introduces many challenges when detecting ground moving objects (GMOs). First, the ground objects are composed of few pixels which makes it difficult to detect these tiny/thin objects from

cluttered environments. Second, the continuous shaky motion of the ACPs makes all objects in the scene appear to be in motion; hence, distinguishing between static objects and moving objects becomes more challenging. Although different recent methods claim to handle the former problems, they suffer from at least one of the following disadvantages: 1) high false detection rates, 2) low true detection rates, or 3) high computational loads. The importance of using ACPs in different critical applications and the absence of a real-time and robust detection method motivated us to investigate the problem of detecting GMOs.

1.3 Research Objective

The objective of this research is to provide a method for robust ground moving objects detection that has the following characteristics:

- Compensate the shaky camera motion of ACPs efficiently with low computational loads
- Achieve a robust detection for small GMOs, i.e. very low false detection rates and high true detection rates.
- Meet the real-time performance requirements

1.4 Contributions Summary

The research presented in this thesis has the following contributions:

1. Proposing robust alignment method with low computational loads to handle the shaky motion of UAVs. The research in this point are published in the following (one journal and two conference papers):

- Agwad ElTantawy and Mohamed S. Shehata. "MARO: matrix rank optimization for the detection of small-size moving objects from aerial camera platforms." *Journal of Signal, Image and Video Processing* 12, no. 4 (2018): 641-649.
- Agwad ElTantawy and Mohamed S. Shehata. "Moving object detection from moving platforms using lagrange multiplier." *In Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 2586-2590. IEEE, 2015.
- Agwad ElTantawy and Mohamed S. Shehata. "UT-MARO: unscented transformation and matrix rank optimization for moving objects detection in aerial imagery." *In International Symposium on Visual Computing*, pp. 275-284. Springer, Cham, 2015. (Oral presentation, acceptance rate 16.5 % - 43/260)

2. Radical reduction of false detection rates and achieving a high true detection rates. To this end, a regularization term is proposed that effectively distinguishes GMOs from false detections. The proposed work on this point results in two (2) novel methods, called KR-MARO and Spring-MARO, that are published in the following (one journal and one conference paper)

   - Agwad ElTantawy and Mohamed S. Shehata. "KRMARO: Aerial Detection of Small-Size Ground Moving Objects Using Kinematic Regularization and Matrix Rank Optimization." *IEEE Transaction on Circuits and System for Video Technology* 2018. DOI: 10.1109/TCSVT.2018.2843761.
   - Agwad ElTantawy, and Mohamed S. Shehata. "A novel method for segmenting moving objects in aerial imagery using matrix recovery and physical spring model." *In Pattern Recognition (ICPR), 2016 23rd International*

Conference on, pp. 3898-3903. IEEE, 2016.

3. Achievement of a real-time detection with low false detection and high true detection rates. This is accomplished via using null space in the detection. Our contributions in this point resulted in 2 methods, LNSP and KR-LNSP, which are being reviewed in the following (two journals)

    - Agwad ElTantawy and Mohamed S. Shehata. "LNSP: Local Null-Space Pursuit for Real-Time Detection in Aerial Surveillance" *Under revision in IEEE transaction on image processing*
    - Agwad ElTantawy and Mohamed S. Shehata. "A Sequential-based PCP method for Ground-Moving Objects Detection from Aerial Videos" *Under revision in Journal of Signal, Image, and Video Processing*

4. Abstract Based Review Papers:

    - Agwad ElTantawy and Mohamed S. Shehata. "Towards Robust Tracking Algorithm" *Annual Newfoundland Electrical and Computer Engineering Conference 2014*
    - Agwad ElTantawy and Mohamed S. Shehata. "Sparse Optimization for Detecting Moving Objects" *Annual Newfoundland Electrical and Computer Engineering Conference 2015*

In summary, the research presented in this thesis has contributed to the body of knowledge through 2 journal papers and 5 conference papers, plus 2 submitted journal papers.

1.5 Thesis Outline

This thesis is organized as follows. **Chapter 2** provides a literature review of moving object detection methods. **Chapter 3** presents how the matrix rank optimization concept is used to address the moving objects detection problem and how to handle the shaky camera motion of UAVs. This is illustrated via the three proposed methods UT-MARO, N-MARO, and I-MARO. **Chapter 4** focuses on reducing false detection. Thus, it demonstrates the regularization term used to accurately distinguish true moving objects from static false detections. **Chapter 5** illustrates effective usage of the null space to achieve the real-time performance. Also, **Chapter 5** demonstrates integration of the regularization term with null space. **Chapter 6** concludes the presented work and depicts future work based upon the research and novel methods proposed.

Chapter 2

Literature Review

The moving objects detection problem has attracted a lot of researchers in the past and remains an area of urgent interest today. Therefore, the literature is rich with proposed detection methods that attempt to tackle different challenges in detecting moving objects [3] [4] [5] [6]. This chapter reviews detection methods that specifically aim to overcome the problems associated with detecting moving objects from an ACP. The main difficulties relate to the relatively small-size of the moving objects, the continuous instability and shakiness of camera motion, the lack of clear differentiating features on the moving objects, and cluttered noisy environments.

Generally, the methods of detecting moving objects from an ACP are categorized into: (1) Background Modelling, (2) Feature-Based Detection, (3) Neural Network-Based Detection, (4) Motion-Based Detection, (5) Different Image Cues for Detection, and (6) Principle Component Pursuit for Detection. The following sections explain each of these categories with examples and attempts to illustrate the advantages and the disadvantages of each category.

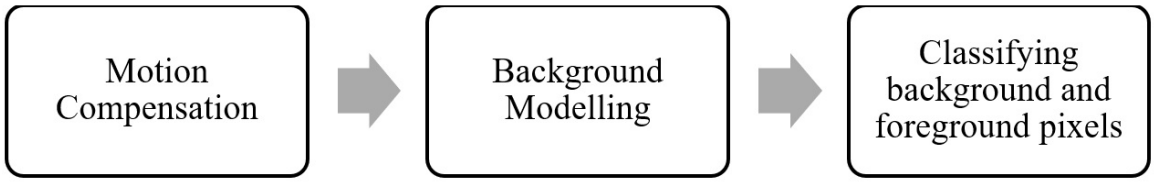


Figure 2.1: The pipeline of background modelling-based detection methods

2.1 GMOs Detection Using Background Modelling

The earliest approach to moving object detection adopted in the field is background modelling. As shown in Figure 2.1, the detection is achieved in three steps: (1) motion compensation, (2) background modelling, and (3) background and foreground pixels classification/segmentation. Motion compensation aims to counter the camera motion in ACPs. This approach involves calculating a transformation matrix that represents misalignment between consecutive frames due to camera motion, and then, uses this matrix to properly align the frames. In general, the transformation matrix is calculated via optical flow across frames [7] or computing the transformation between sets of matched features in consecutive frames [8]. Background modelling is a process of constructing a representation of the background (called background model). The background model is built from the background in previous frames. Classifying the background and foreground (i.e. moving object) pixels is achieved by differentiating and obtaining the relative variation of the model and input frame. The variation is significant at foreground pixels, while the variation is very low in the background pixels.

Generally, background modelling-based detection methods are grouped according to the methods used to construct the background model: i.e., non-recursive, recursive, and panoramic background.

2.1.1 Non-recursive methods:

Non-recursive methods use previous n frames as a background model for the input frame. Then, temporal variance between the input frame and the background model is used to detect the moving objects. Typically, a pixel in the input frame is considered as a foreground when its intensity value is greater than or less than the corresponding pixel(s) in the background model.

Irani et al., in [7], use previous frame as a background model for input frame. Subtracting input frame from the background model turns the background pixels to zeros while retaining the moving objects' pixels. In COCOA [8], a set of frames is considered as the background model for input frame. Then, the moving objects are detected using an accumulative frame difference between the background model and input frame. Due to the simplicity of Non-recursive methods, they are sensitive to illumination changes, which is common in outdoor environments [9].

2.1.2 Recursive methods:

Instead of directly using previous frames as the background model, recursive methods uses statistical analysis to build the background model. Each pixel in the background is represented by statistical parameters (e.g. mean and variance, etc.) and these parameters are updated for each input frame. Hence, Recursive methods become more reliable against illumination changes.

For example, Wren et al. [10] use a Gaussian model to build the background model where each background pixel is represented by mean and variance. Next, the pixels in an input frame are classified into background and moving object pixels based on the following rules: If the intensity value of a pixel is outside the variance of its corresponding pixel in the background model, then this pixel belongs to a moving

object. However, if a pixel's intensity is within the variance of its corresponding pixel in the background model, then this pixel is a background pixel and it will be used to update the mean and the variance in the background model. To model complex background, Hayman et al. [11] use a mixture of Gaussian [12], while Zivkovic et al. [13] propose an adaptive mixture of Gaussian method. In [14], a double Gaussian model is applied to allow for greater adaptability for outdoor scenes.

The main drawback of recursive methods is that any error in the background model can linger for a much longer period of time. Hence, the errors introduced by the camera motion in ACPs corrupts the background model [9]. Although typical recursive methods apply a motion compensation to handle the camera motion, in practice, it is often difficult to ensure the accuracy and reliability of motion compensation techniques.

2.1.3 Panoramic background methods:

These methods (e.g. [15]) use multiple images of the scene that are free from the moving objects to construct a mosaic background [16]. This mosaic background is employed as the background model. With this approach moving objects are detected by comparing the mosaic background with the current frame. However, any turbulence in the illumination or intensities between the mosaic background and the current frame leads to significant false detection rates [9].

2.2 Feature-based Detection

Figure 2.2 shows the general steps to detect moving objects using feature-based detection methods. First, a set of interest/feature points (e.g. SIFT features, corner points, etc.) are detected and matched between two frames. Then, the matched fea-

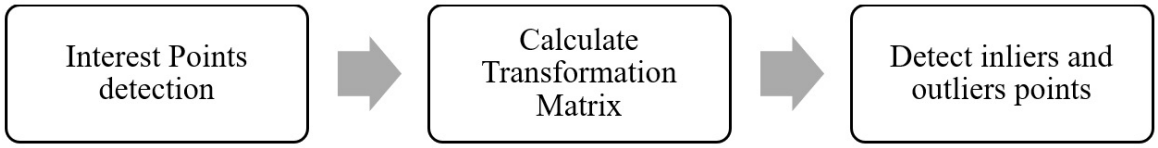


Figure 2.2: The pipeline of feature-based detection methods

tures are used to calculate a transformation matrix that maps the latter frame to the former frame. The matched features that cannot be mapped using the transformation matrix are designated outliers and they represent points on the moving objects. On the other side, the matched features that can be mapped using the transformation matrix are inliers and represent background points.

Ollero et al., in COMETS [17], uses corner points [18] as interest points. The corner points are matched between a pair of frames to calculate a homography matrix that maps the corner points in a latter frame into their corresponding locations in a former frame. Then, moving objects are detected using outlier points that cannot be mapped using the homography matrix. In [19], FAST features [20] are matched between consecutive frames and used to calculate a homography matrix that represents the camera motion between frames. Next, the Least Median Square Estimator (LMedS) algorithm [21] is used to classify FAST features into outlier and inlier. Finally, inlier points are clustered to form moving objects. In [22], Li et al. propose using a Multiview-based Parameter Free (MPF) framework to detect moving objects (or groups of moving people in crowded scenes) by tracking a set of feature points over time. These points are grouped according to motion and context similarity into different moving objects and differentiated from background. Xu et al. [23] enhance the detection accuracy by proposing a framework that selects certain matched features for use in calculating the transformation matrix.

Feature-based methods can attain a high accuracy under the assumption that each moving object has a subset of the interest points. However, in the case of aerial

imagery, this assumption is not always valid due to the small size of the moving objects and the low contrast/blurring of the background [24].

2.3 Neural Network-based Detection

Instead of using traditional features (a.k.a. hand crafted features, e.g. FAST, HOG, etc.), Convolutional Neural Network (CNN) [25] generates features to accurately describe the regions of moving objects and background regions. For example, in R-CNN [26], Girshick et al. use a CNN to extract features from pre-selected regions, then classify these regions using a SVM classifier [27]. Sermanet et al. propose overfeat method [28] that comprises two CNNs. The first CNN detects regions that contain the moving objects, while the second CNN estimates the bounding box of these objects.

However, CNN-based methods suffer from the class imbalance problem [29] that drastically reduces their performance. This problem occurs when the number of negative classes dominate the positive classes. Therefore, CNN provides an accurate detection when the moving objects occupy a large part of the scene (e.g. ImageNet dataset [30]). Yet, in the case of aerial imagery, the background (the negative class) is the dominant part, while the moving objects (the positive class) are quite small (you may have only one small object in a scene). As reported in [31], when testing the CNN-based detection method [32] on aerial imagery, the detection accuracy drops radically, i.e. the true positive rate (TPR) is 40 % on average. Tayara et al. in [33] enhance the detection accuracy, i.e. TPR is 90.51 % when testing on the datasets [34] and [35]. However, their method is limited to detecting a specific class of moving objects(in the case vehicles) and requires high resolution images. Other techniques attempt to solve the class imbalance problem, e.g. though data augmentation [36],

however, they still cannot reach a high accuracy when augmenting few classes [37].

2.4 Motion-Based Detection

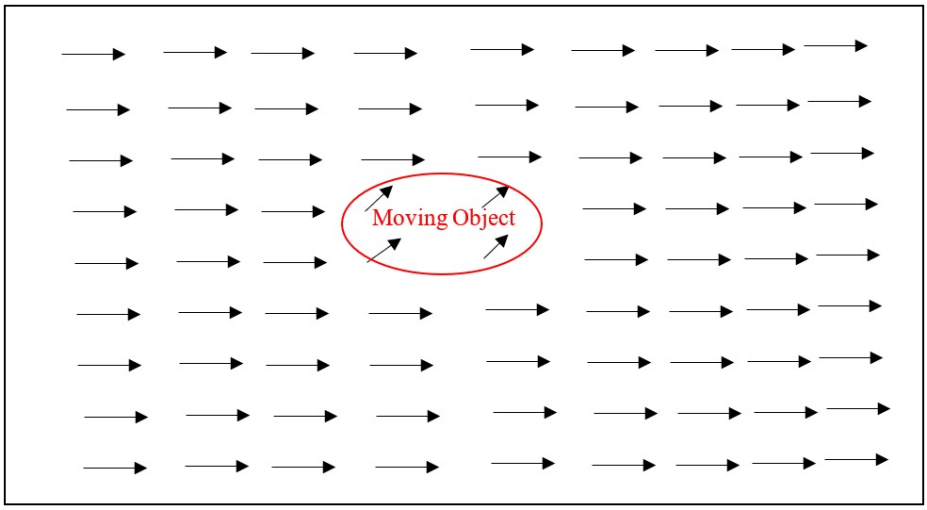


Figure 2.3: Example of using optical flow to detect a moving object

The general concept in motion-based detection [38] is to calculate a motion vector for each pixel in the input frame using optical flow; then, group the pixels according to their motion vectors into either moving object pixels or background pixels. The motion vectors of background pixels satisfy the following three conditions:

- They are aligned with the global motion in the frame
- They have similar magnitude and direction
- They represent the majority in input frame

Any motion vector that does not meet the former conditions is for a pixel on a moving object. Figure 2.3 shows an example of detecting moving objects using optical flow.

Aslani et al. [39] implement the former concept by calculating motion vectors for the input frame's pixels using [40]; next, the moving objects are identified according to the obtained motion vectors. Yokoyama et al. [41] use motion vectors and an object's contour to detect moving objects.

Instead of calculating the motion vector in one frame, Brox and Malik [42] propose calculating motion vector over time to create long term pixel trajectories. Then, spectral clustering is used to group these trajectories into background and moving object trajectories. Similarly, Singh et al. [43] calculate pixel trajectories, but use a bag of words classifier to detect moving objects' trajectories.

Unfortunately, calculating the optical flow in motion-based detection methods is the key failure of using these methods in detecting moving objects from ACPs. The pioneer optical flow method [40], called Lukas Kanade, suffers from errors in determining moving object boundaries [44]. Therefore, many errors are expected on the edges, as the whole scene in aerial imagery is moving due to the camera motion. To clarify this point, Lukas Kanade method is applied on frame 20 and 21 from UCF aerial action 2, shown in Figure 2.4. From ellipses 4 and 5 in Figure 2.4, Lucas Kanade method fails to detect the optical flow vectors correctly. Although, the background motion is rigid, ellipses 1, 2, and 3, they have different optical flow vectors. This is due to the existence of edges in these regions. The actual flow vectors of the moving objects are marked by the red arrow in Figure 2.5.

Although the problem with Lucas Kanade method problem is solved in the Horn and Schunk method [45], smoothness and data terms in Horn and Schunk do not allow for discontinuities in the optical flow field to handle outliers robustly [46]. As shown in Figure 2.6, the edges problem which characterizes and limits the Lucas

Kanade method disappears in Horn and Schunk method. Nevertheless, it is hard to distinguish the moving objects from that the resulting optical flow; as shown in ellipses 1, 2, and 3 in Figure 2.6, motion vectors of moving objects are quite similar to the motion vectors in the background pixels

Most of these drawbacks are successfully treated in modern optical flow methods (e.g. [47]), however, their coarse-to-fine behaviour makes them inefficient when detecting small or thin moving objects, such as in aerial imagery, [48]. An example of coarse-to-fine optical flow methods is Classic-NL [49] (one of the modern and efficient techniques for calculating optical flow) which down-samples an image into different smaller scales, then calculates the optical flow from the smallest scale (coarse level) to the original scale of the image (fine level). This down-sampling of the image wipes out small or thin objects. For example, in Figure 2.7, the motion vectors in ellipses 1, 2, and 3 are different; however all these regions belong to the background and they should have similar motion vectors. In ellipse 1 in Figure 2.7, angles of the optical flow vectors are almost horizontal, while in ellipse 3 Figure (2.7), they are 45 degrees. In ellipse 2 Figure 2.7, the angles are approximately 20 degrees.

Another general disadvantage of optical flow is the difference between motion vectors and the physical motion field. For example, the physical motion field of the barber's pole is horizontal while the optical flow is vertical, as in Figure 2.8. This can be observed in aerial imagery as well, in Figure 2.9.

2.5 Different Image Cues for Detection

In contrast to the former approaches described above that depend on one cue from the frames, recently, researchers tend to merge different cues to improve detection accuracy. Earlier attempts used spatial and temporal cues. Usually, the temporal

information is extracted using either background modelling or optical flow to provide an initial detection of the moving objects in consecutive frames. After that, the spatial information, e.g. saliency, is used to filter the results. Shen et al. [50] use frame difference with motion compensation to detect the moving objects, then pixel and region saliency are used to filter the output. Pouzet et al. [24] propose to use a spatio-temporal tracker after compensating the motion of the camera. Although these methods may reduce the false detections, they still are not robust with small size moving objects and moving camera platforms due to using background modelling or optical flow in the initial detection.

2.6 Principle Component Pursuit for Detection

The basic idea of PCP is proposed by Candes et al. in [51] where $\ell_* - norm$ and $\ell_1 - norm$ can be used to obtain a low-rank matrix (core structure) and identify noise in an corrupted observation matrix. Zhou et al. in [52] propose a relaxed PCP to adapt to environmental noise. In TFOCS [53], $l_\infty - norm$ is used to capture the quantization error in the observation matrix.

Recently, PCP (i.e., Robust Principal Component Analysis) has been extensively used to resolve several of the main problems that arise in moving objects detection. Adopting PCP into the detection is based on the assumption that a background in a video can be viewed as the video's core structure; hence, the background can be modelled as a low-rank matrix which is obtained via $\ell_* - norm$. Due to the motion of the moving objects, they can be modelled as sparse corrupt the video and obtained via $\ell_1 - norm$. PCP provides more robust and accurate detection in different challenging indoor and outdoor environments [54]. Moreover, PCP shows great success in detecting small moving objects [55]. Generally, PCP-based detection methods are

divided into two groups: batch-based PCP detection and sequential PCP detection. In batch-based PCP detection, a set of frames has to be buffered before performing any detection. In contrast, sequential PCP detection methods can perform detection without buffering frames. The following subsections illustrate different methods for both batch-based PCP detection and sequential PCP detection methods showing the evolution of using PCP methods in moving object detection.

2.6.1 Batch-based Principle Component Pursuit

Stationary Camera

In [56], the moving objects are detected by using $\ell_1 - norm$, an approach also used by Candes et al. Additionally, to reduce the rate of false detections, the moving objects are penalized based on their temporal connectivity using a total variation penalty. Instead of using the temporal connectivity, Xin et al. [57] assume that moving objects should correspond to meaningful shapes, such as human, cars, etc. For this purpose, a Fused Lasso term is added to the minimization problem to penalize the spatial connectivity between the pixels of moving objects. Rezaei et al. [58] use objectness detection to penalize the moving objects. The objectness detection targets to score the moving objects according to their salient properties that are obtained via Edge boxes [59].

Tang et al. [60] achieve the detection of moving objects by introducing a new term, i.e., $l_{2,1} - norm$ in the Candes et al. formulation. This $l_{2,1} - norm$ ensures that the columns of the low rank matrix are zeros at the location of the moving objects. Zhu et al. [61], use $l_{2,1} - norm$ and spatial connectivity term to penalize the moving objects. In [62], the total variation penalty with $l_{2,1} - norm$ is used to detect moving objects with low false detections rates.

Although these former PCP-based detection methods achieve a robust detection, they do not consider the camera motion. Moreover, they penalize moving objects based on spatial information or temporal connectivity, and ignore the motion cues of these objects (e.g. motion pattern, displacement, etc.). Hence, these methods are not suitable for the main problem investigated in this research, which is detecting moving objects from moving ACPs.

Moving Camera

To handle camera motion, RASL-APG [63] formulates the detection problem using PCP with a transformation domain which represents the misalignment between successive frames. RASL-APG solves this formulation using the accelerated proximal gradient (APG) algorithm [64]. In [65], Peng el al. propose RASL-IALM to enhance their previous work (i.e. RASL-APG) by using IALM to solve the same formulation. Both RASL-APG and RASL-IALM use the Newton method [66] to calculate the transformation domain. DECOLOR [54] utilizes the SOFT-IMPUTE algorithm [67] to solve RASL-APG's problem formulation, and employs the weighted-Newton method [68] for better calculation of the transformation domain. Although DECOLOR reduces false detections, it is at the expense of true detections.

3TD [55] use PCP and an object confidence map to penalize the moving objects based on their motion cue. The object confidence map is constructed by using the dense optical flow method [69] to obtain the motion pattern of each pixel. Then, the object confidence map assigns a weight to each pixel according to its motion pattern. Finally, the weights of these pixels are used to filter out false detections. Similarly, Gao et al. [70] propose the Block-Sparse RPCA method, but instead of calculating the dense optical flow, Block-sparse RPCA calculates the optical flow Classic-NL [49] only in the regions that may contain moving objects (we refer to these regions with

candidate moving objects as $CMOs$). First, Block-sparse RPCA uses RASL-IALM to get initial detections of all $CMOs$, then applies the Classic+NL optical flow [49] to create a saliency map. This saliency map is integrated into the PCP formulation to detect the moving objects. However, coarse-to-fine behaviour of used optical flow methods in 3TD and Block-sparse RPCA (i.e. dense optical flow method [69] and Classic-NL [49]) their coarse-to-fine behaviour makes them inefficient to detect small or thin moving objects, such as in aerial imagery, [48].

2.6.2 Sequential-based Principle Component Pursuit

The computational loads of batch processing nature in the above PCP detection methods motivate many researchers to investigate handling of the computational load drawback and reaching a real-time performance. Therefore, different sequential PCP methods are proposed. The first attempt is GROUSE [71] which models the background as a subspace that lies in a low dimension subspace; Next, the background is estimated at each time based on this low dimension subspace. Typically, GROUSE recovers the background by minimizing the gradient of the $\ell 2 - norm$ cost function between a video frame and the low dimension subspace, which is the orthonormal matrix. Since the background is an evolving subspace, the orthonormal matrix is updated based on the Grassmannian manifold. GRASTA [72] extends GROUSE in two key ways: (1) modelling the moving objects as sparse corrupting a video frame which can be detected by minimizing $\ell_1 - norm$, and (2) reducing the sensitivity to the noise by computing a gradient of the $\ell 1 - norm$ cost function. ROSETA [73] adds a smoothness term to maintain the proximity of the orthonormal matrix current estimate to its previous estimate, and it also adds an adaptive parameter selection strategy to reach precise results with low computational overhead. Improved ROSETA [74] includes an $\ell 2 - norm$ data misfit term and an $\ell 1 - norm$ regularization term applied to the

outliers. Although GRASTA, ROSETA and improved ROSETA achieve adequate real-time performance, they suffer from high false detections when used for aerial videos. Xu et al., in GOSUS [75], enhance this performance via object contiguity by penalizing the moving objects, but they fail to reach adequate real-time performance.

Instead of using the orthonormal matrix, Qui et al., in ReProCS [76], calculate the null space [77] of the background in a set of frames, then project this null space into the current frame to nullify the background and retain the moving objects. PracReProCS-pPCA [78] extends ReProCS by introducing a slow subspace change where the subspace is estimated every n frames to adapt to the background changes. However, PracReProCS-pPCA cannot cope with the rapid background changes caused by the shaky camera motion in aerial videos. Moreover, ReProCS and PracReProCS-pPCA are computationally intensive methods.

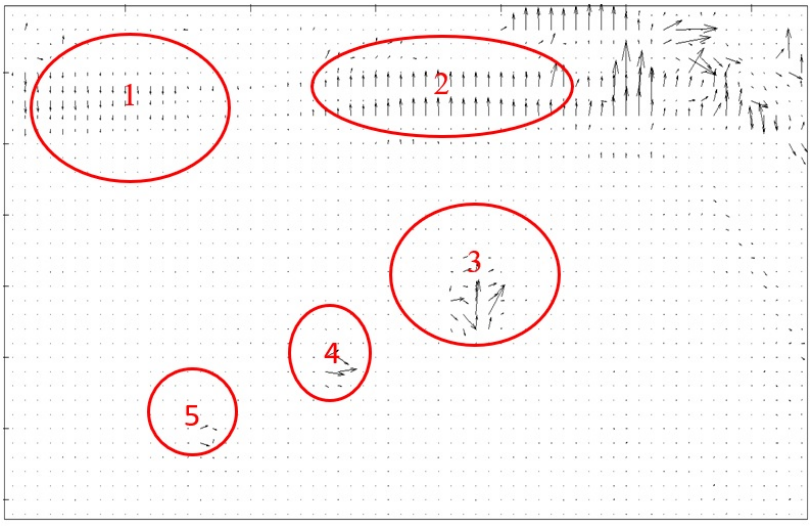
amFastPCP [79] replaces the constraints of the original PCP formulation [51] with penalties to the subspace rank and vice versa. Then, the background is estimated sequentially using a partial SVD, while the moving objects are detected using a shrinkage of the current frame. Rodriguez et al., in incPCP [80], reduce the computational and memory loads by proposing thin SVD. Rodriguez et al. in [81] overcome the ghost effect of amFastPCP and incPCP by substituting the standard $\ell 1 - norm$ with a projection onto the $\ell 1 - ball$. Nevertheless, the performance of the method in [81], incPCP and amFastPCP drops drastically in aerial videos because of the camera motion.

To handle the camera motion in aerial videos, Chau et al., in [82], introduce a transformation domain to align the current background estimate with the previous backgrounds estimates; however, their method is computationally expensive. Shakeri et al. propose COROLA [83] to handle both the camera motion and reduce the computational loads. COROLA avails of the ideas in the DECOLOR [54] and OR-

PCA [84] methods. Although, COROLA outperforms other S-PCP methods and reduces the computational loads, it suffers from ghosts effects. The resulting errors from this effect aggregate over time until the whole frame is detected as a single moving object. Moreover, it does not meet real-time requirements.

UCF aerial action 2, Frame 20

UCF aerial action 2, Frame 21

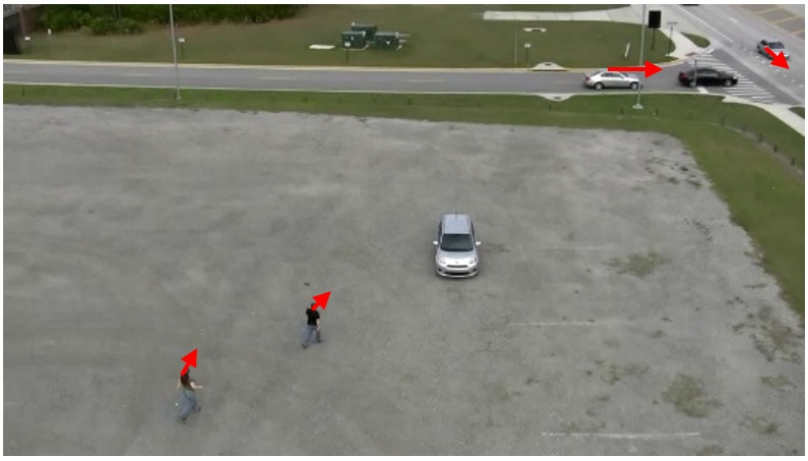


Motion vectors using Lukas Kanade method

Figure 2.4: Example of applying Lukas Kanade optical flow in detecting a moving object

UCF aerial action 2, Frame 20

UCF aerial action 2, Frame 21

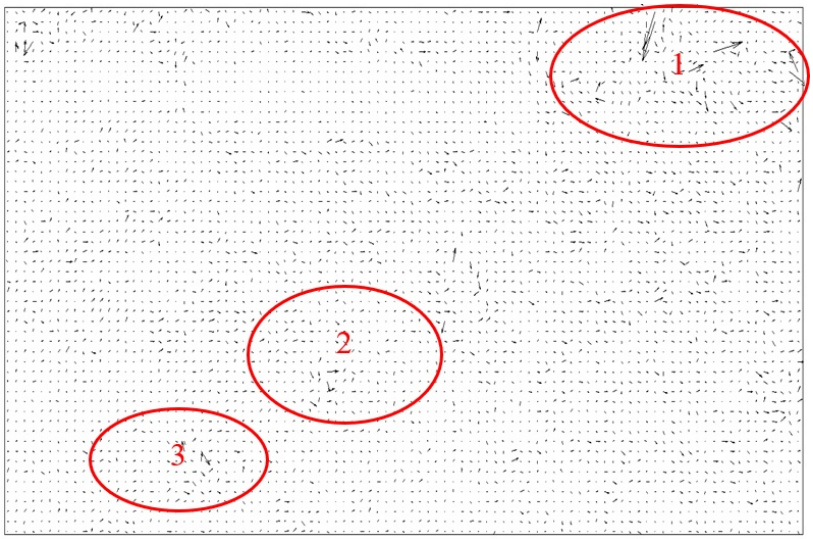


True Motion vectors of the motion objects

Figure 2.5: Actual motion vectors for the moving objects in Frame 20 and 21 in UCF aerial action

UCF aerial action 2, Frame 20

UCF aerial action 2, Frame 21

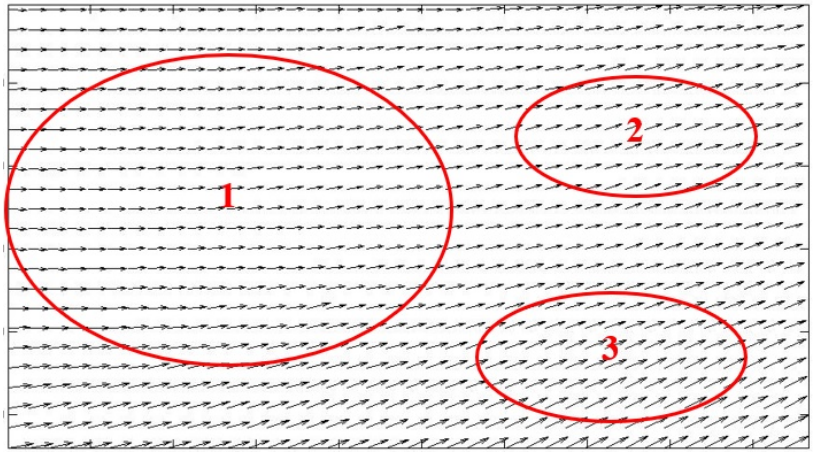


Motion vectors using Horn and Schunk method

Figure 2.6: Example of applying Horn and Schunk optical flow in detecting moving objects

UCF aerial action 2, Frame 20

UCF aerial action 2, Frame 21



Motion vectors using Classic-NL method

Figure 2.7: Example of appying Classic-NL optical flow in detecting moving objects

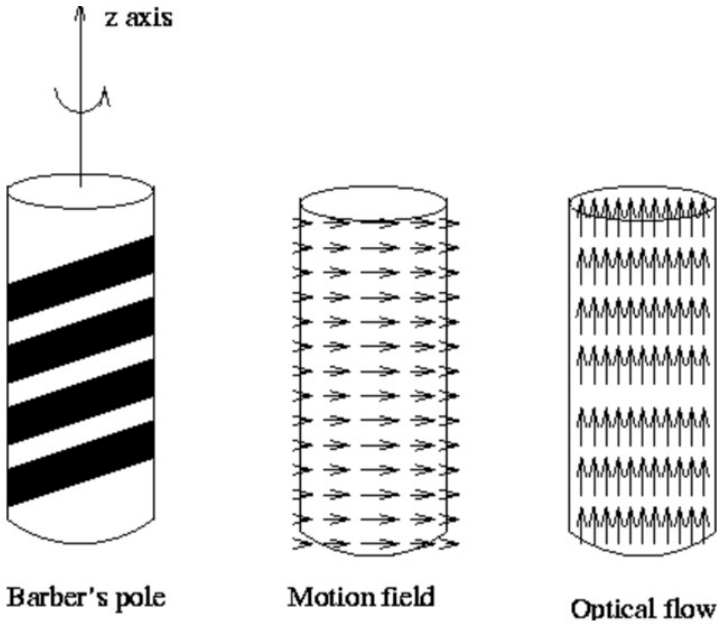
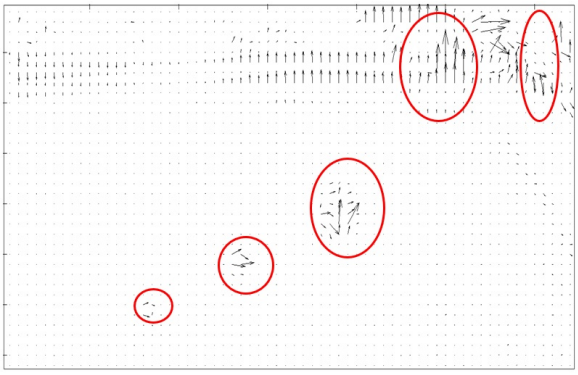


Figure 2.8: Motion vectors of moving barber's pole

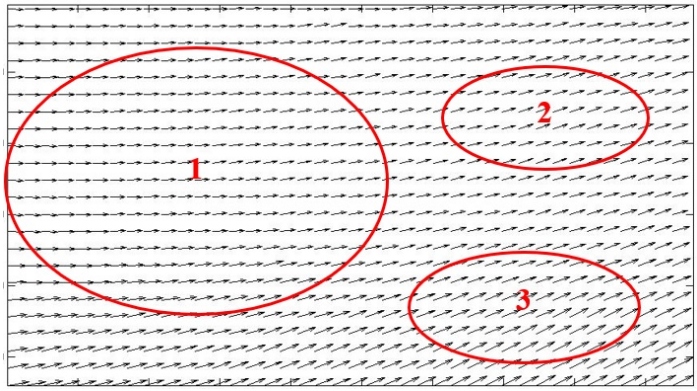
UCF aerial action 2, Frame 20

UCF aerial action 2, Frame 21



Lucas Kanade method

Horn and Schunk method



Classic-NL method

Figure 2.9: Motion vectors in aerial images

Chapter 3

Moving Objects Detection Using Principle Component Pursuit

3.1 Introduction

The problem of retrieving the core structure of corrupted observations attracts a lot of attention in different research fields [85], e.g. matrix completion, computer vision, machine learning, etc. The first attempts in this direction are based on Principal Component Analysis (PCA) [86] in which the core structure of the corrupted observation is modelled as a low-rank matrix; hence, PCA searches for the best rank-k estimate of this low-rank matrix that minimizes the $\ell_2 - norm$ between the observation and this estimate. Although, PCA has been widely deployed in a diverse range of applications, PCA is not suitable for use in modern applications that deal with grossly corrupted observations. Recently, Principal component pursuit (PCP) [51] has been proposed as a robust alternative for PCA. In PCP, the observation is decomposed into two components: a low-rank component (the core structure) and a sparse component by minimizing a weighted combination of the nuclear norm and the $\ell_1 - norm$.

3.2 Chapter Contributions

This chapter illustrates how PCP is utilized to formulate the problem of detecting moving objects. Since the main scope of this thesis is detecting moving objects from ACPs, this chapter focuses on handling camera motion in ACPs. To this end, three detection methods are proposed: (1) Unscented Transformation with MAtrix Rank Optimization (UT-MARO), (2) MAtrix Rank Optimization with weighted Newton (N-MARO), and (3) MAtrix Rank Optimization with inexact Newton method (I-MARO). These methods are published in one journal and two conferences:

- Agwad ElTantawy and Mohamed S. Shehata. "MARO: matrix rank optimization for the detection of small-size moving objects from aerial camera platforms." *Journal of Signal, Image and Video Processing* 12, no. 4 (2018): 641-649.

- Agwad ElTantawy and Mohamed S. Shehata. "Moving object detection from moving platforms using lagrange multiplier." *In Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 2586-2590. IEEE, 2015.

- Agwad ElTantawy and Mohamed S. Shehata. "UT-MARO: unscented transformation and matrix rank optimization for moving objects detection in aerial imagery." *In International Symposium on Visual Computing*, pp. 275-284. Springer, Cham, 2015. (Oral presentation, acceptance rate 16.5 % - 43/260)

The results in this chapter are replicated from these former publications.

3.3 Principal Component Pursuit for Moving Object Detection

Given a sequence of frames, the goal is to extract the moving objects from these frames. Ideally, the background intensity in these frames should not change except for variations caused by illumination changes. Consequently, the background in the frames is linearly correlated, which implies that this background can be modelled as a low-rank matrix. However, the existence of moving objects, the motion of which differs from that of the background, causes an interruption in the background intensity. This interruption cannot fit into the low-rank model of the background. Therefore, the moving objects can be modelled as sparse or outliers of the low-rank matrix.

From the above, the sequence of frames can be viewed as a combination of two components: low-rank matrix (the background) and sparse (the moving object). To detect the moving objects from a sequence of frames, the sequence should be decomposed into its components. Mathematically, this decomposition can be achieved by solving the following PCP formulation:

$$\min_{B,O} \; Rank(B) \quad s.t. \;\; F = B + O, \;\; \|O\|_0 \leq \varepsilon \tag{3.1}$$

where $F \in \mathbb{R}^{n \times m}$ is a frames matrix that is defined as $F = [vec(f_1), ..., vec(f_m)]$; n is the resolution of a frame $f_i \in \mathbb{R}^{w \times h}$; m is the number of frames in F; $vec : \mathbb{R}^{w \times h} \rightarrow \mathbb{R}^n$; $B \in \mathbb{R}^{n \times m}$ denotes the background in the frames matrix; $O \in \mathbb{R}^{n \times m}$ represents the moving objects; $Rank(.)$ signifies the rank of a matrix (matrix rank is the maximum number of linearly independent column vectors in the matrix); $\|.\|_0$ denotes $L_0 - norm$ which is as a total number of non-zero elements in a matrix; ε is a constant that represents the maximum number of corrupted pixels by the moving

objects. A more convenient form of Eq. 3.1 can be rewritten in the following Lagrange form:

$$\min_{B,O} \; Rank(B) + \lambda\|O\|_0 \quad s.t. \;\; F = B + O \tag{3.2}$$

where λ is a weighting parameter.

However, optimizing Eq. 3.2 with the existence of the non-convex functions $Rank(.)$ and $\|\|_0$ is not tractable. Fortunately, a convex relaxation can be applied under the assumption that the rank of B is not too high and the number of non-zero entries in O is small [51]. Hence, the nuclear norm $\|B\|_*$ and the $\ell_1 - norm$ $\|O\|_1$ can be used as the natural surrogates for $Rank(B)$ and $\|O\|_0$, respectively to recover B and O:

$$\min_{B,O} \; \|B\|_* + \lambda\|O\|_1 \quad s.t. \;\; F = B + O \tag{3.3}$$

$\|.\|_*$ is defined as the sum of singular values in a matrix. $\|.\|_1$ the sum of all elements in a matrix.

The following sections illustrate the solution of the moving objects detection problem formulation in Eq. 3.3 via three proposed novel methods: UT-MARO, N-MARO, and I-MARO. Additionally, each of these methods provides a unique way of handling the camera motion in ACPs.

3.4 UT-MARO

The continuous motion of ACPs represents the main challenge for any such detection method, as distinguishing between real moving objects and static background elements (which will appear as moving) presents a number of unique technical difficulties. This is similar to the case of using PCP formulation to detect moving objects from ACPs. The camera motion in ACPs makes the similar background parts across the frames to

be misaligned; hence, it puts the validity of modelling the background as a low-rank in jeopardy. Therefore, UT-MARO proposes novel feature-based frame alignment method to handle the camera motion problem. This alignment method utilizes the unscented transformation [87] to calculate a transformation domain T that represents the misalignment between consecutive frames ($T \in \mathbb{R}^{3\times3\times m}$).

3.4.1 Unscented Transformation for Frame Alignment

Most PCP-based detection methods found in the literature, such as RASL-IALM, use the Newton method to calculate the misalignment between frames. However, estimating the misalignment using the Newton method is more complicated than using the unscented transformation with a probability distribution approximation [87]. Typically, the unscented transformation uses a set of points, called sigma points $\{\sigma\}$, that encode the statistical properties of the input space. Then, these points are propagated into the non-linear system to obtain a cloud of transformed points.

UT-MARO calculates $T = \{t_1, ..., t_m\}$ (t_i is an affine transformation matrix for frame f_i) by propagating a set of matched points between the frames in F through the unscented transformation, as show in Figure 3.1. This is achieved via the following steps:

1. Set the first frame in F as a reference frame f_{ref}
2. Detect SURF feature points [88] in f_{ref}
3. Detect SURF feature points in frame f_i
4. Match these feature points with the corresponding SURF feature points in f_{ref}. $Mtch_{ref}$ (input space) represents the corresponding matches in f_{ref}, while $Mtch_i$ (the transformed points) denotes the matched SURF feature points in f_i.

5. Compute the statistical properties, i.e. $Mean$ and the covariance matrix Σ, for $Mtch_i$ and $Mtch_{ref}$.

6. Calculate five sigma points for $Mtch_i$ and $Mtch_{ref}$ as follows:

$$\sigma_{i,1} = Mean_i \tag{3.4}$$

$$\sigma_{i,j} = \sigma_{i,1} \pm (\sqrt{(dim)}\Sigma_i), \quad j = 2, 3, 4, 5 \tag{3.5}$$

where where $\sigma_{i,j}$ denotes a sigma point for $Mtch_i$; dim is matched feature points dimensions. The sigma points $\{\sigma_{ref,1}, ..., \sigma_{ref,5}\}$ of $Mtch_{ref}$ are calculated same as in Eq. 3.4 and Eq. 3.5.

7. Calculate affine transformation matrix $(t^{'}_{i,j})$ for each corresponding pair of the sigma points $\sigma_{i,j}$ and $\sigma_{ref,j}$. Since the minimum number of points required for calculating the affine transformation matrix is four points, four extra sigma points are calculated for both $\sigma_{i,j}$ and $\sigma_{ref,j}$. These extra sigma points are calculated as follows:

$$\sigma^{'}_{i,j,k} = Mean^{'}_{i,j} \pm (\sqrt{(dim)}\Sigma_i), \quad k = 1, 2, 3, 4 \tag{3.6}$$

where $\sigma^{'}_{i,j,k}$ denotes an extra sigma point for $\sigma_{i,j}$; $Mean^{'}_{i,j} = \sigma_{i,j}$.

8. Calculate the affine transformation matrix t_i that maps f_i to f_{ref} as a weighted average of the five transformation matrices $\{t^{'}_{i,1}, ..., t^{'}_{i,5}\}$, as shown in the following equation:

$$t_i = \frac{1}{5}\sum_{i=0}^{5} w_i t^{'}_{i,j} \tag{3.7}$$

where $w_{i,j}$ are weight calculated as:

$$w_{i,1} = \frac{\xi}{(d+\xi)} \tag{3.8}$$

$$w_{i,j} = \frac{\xi}{(dim+\xi)}, \quad j = 2,3,4,5 \tag{3.9}$$

where ξ is a scaling parameter.

9. Update f_{ref} by f_i if the transformation matrix t_i is greater than a certain threshold. This will help adapt to compensate for high shakiness in new frames after f_i.

10. Repeat steps 2-9 for f_{i+1} to obtain t_{i+1}

After calculating the transformation domain T, The frames in F are aligned as in the following Equation:

$$\tilde{F} = F \bullet T \tag{3.10}$$

where $\tilde{F}$ is a matrix of aligned frames, $\bullet$ is an alignment operator.

3.4.2 Video Decomposition

Given the alignment operation is applied on the frames in F, then, the problem formulation in Eq. 3.3 can be re-written as:

$$\min_{B,O} \ \|B\|_* + \lambda\|O\|_1 \quad s.t. \ \ \tilde{F} = B + O \tag{3.11}$$

Consequently, to detect the moving objects, UT-MARO decomposes $\tilde{F}$ into moving objects and background. The decomposition is obtained via Inexact Augmented Lagrange Multiplier method (IALM). IALM is a type of Augmented Lagrange Multiplier

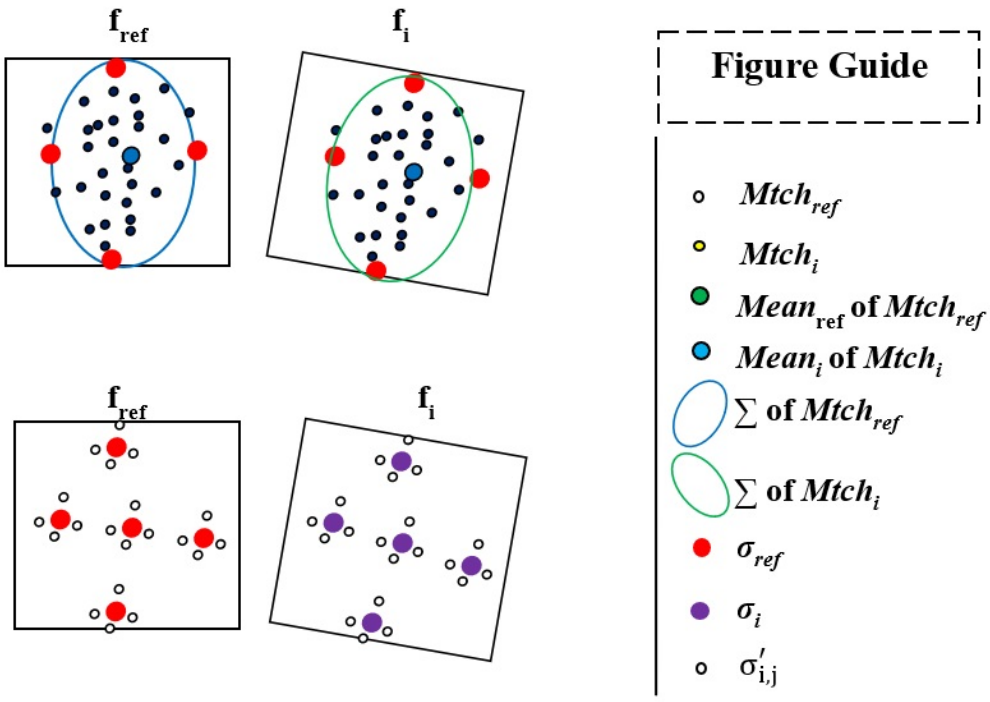


Figure 3.1: Unscented Transformation of Feature Points

solver that is designed to provide an accurate and fast solution to PCP problems. It is five times faster than other PCP solvers with a higher precision. IALM replaces the constrained optimization problem in Eq. 3.11 with a series of unconstrained problems and adds a penalty term to the objective function, as follows:

$$\mathcal{L}(B, O) = \|B\|_* + \lambda\|O\|_1 + \langle Y, \tilde{F} - B - O\rangle + \frac{\mu}{2}\|\tilde{F} - B - O\|_F^2 \tag{3.12}$$

where μ is a positive scalar; $\|.\|_F^2$ is the Frobenius norm; and $\langle X, Y\rangle$ denotes the inner product. The Lagrange multiplier Y is calculated as:

$$Y \leftarrow Y + \mu(\tilde{F} - B - O) \tag{3.13}$$

Then, IALM chooses the best μ to reduce the number of singular value decompositions (SVD) required to obtain the low-rank matrix and obtain accurate values for B and O. Finally, an alternating strategy is adopted in which IALM minimizes Eq. 3.12 iteratively with respect to its components, i.e. B and O:

$$B_{k+1} = arg\min_{B} \mathcal{L}(B, O_k) \tag{3.14}$$

$$O_{k+1} = arg\min_{O} \mathcal{L}(B_k, O) \tag{3.15}$$

In the result of this section, the closed-form solutions of B and O are derived from Eq. 3.14 and Eq. 3.15. For simplicity, the subscript k in these equations is dropped when deriving the closed-form solutions. Also, the expected terms to be zeros due the partial derivative are removed from the beginning. For example, when obtaining the closed-form solution of B from Eq. 3.14, terms $\|O\|_1$, $\langle Y, \tilde{F}$, and $\langle Y, O\rangle$ are turned to be zeros since the partial derivative is over B.

Closed-form Solution of B

From Eq. 3.12, Eq. 3.14 can be written as follows:

$$B = arg\min_{B} \|B\|_* + \langle Y, -B\rangle + \frac{\mu}{2}\|\tilde{F} - B - O\|_F^2 \tag{3.16}$$

Since, $\|x\|_f^2 = \langle X, X\rangle$, then we replace the Frobenius norm with an inner product, expand the inner product and separate B:

$$B = arg\min_{B} \|B\|_* + \langle Y, -B\rangle + \frac{\mu}{2}\|B\|_F^2 - 2\langle \tilde{F} - O, B\rangle \tag{3.17}$$

The former equation can be simplified as:

$$B = \underset{B}{arg\min} \|B\|_* + \frac{\mu}{2}\|B\|_f^2 - 2\langle \tilde{F} + \mu^{-1}Y - O, B\rangle \tag{3.18}$$

Since, $\|B\|_f^2 = \langle B, B\rangle$, then $\langle \tilde{F} + \mu^{-1}Y - O, B\rangle$ can be combined with $\|B\|_f^2$ into a single Frobenius norm:

$$B = \underset{B}{arg\min} \|B\|_* + \frac{\mu}{2}\|B - \tilde{F} + \mu^{-1}Y - O\|_F^2 \tag{3.19}$$

By applying the single value thresholding algorithm [89], the closed-form solution is

$$B = U\mathcal{S}_{\mu^{-1}}(\Sigma)V^T \tag{3.20}$$

where U, Σ and V are SVD of the term $(\tilde{F} + \mu^{-1}Y - O)$; $\mathcal{S}_\alpha(x)$ is the soft thresholding operator defined as in Eq. 3.21

$$\mathcal{S}_\alpha(x) = sign(x) \times max\{|x| - \alpha, 0\} \tag{3.21}$$

Closed-form Solution of O

From Eq. 3.12, Eq. 3.15 can be written as follows:

$$O = \underset{O}{arg\min} \frac{\lambda}{\mu}\|O\|_1 + \frac{1}{2}\|O - (\tilde{F} + \mu^{-1}Y - B)\|_f^2 \tag{3.22}$$

Since the $\|O\|_1$ function is non-differentiable, then, subdifferential ∂ of Eq. 3.22 is obtained and set to zero [90], as follows:

$$0 = \frac{\lambda}{\mu}sign(O) + O - (\tilde{F} + \mu^{-1}Y - B) \tag{3.23}$$

rearrange the terms

$$O = \frac{\lambda}{\mu} sign(O) + \tilde{F} + \mu^{-1}Y - B \tag{3.24}$$

and this can be expressed as:

$$O = \begin{cases} \tilde{F} + \mu^{-1}Y - B - \frac{\lambda}{\mu}, & \textbf{if} \quad \tilde{F} + \mu^{-1}Y - B > \frac{\lambda}{\mu} \\ \tilde{F} + \mu^{-1}Y - B + \frac{\lambda}{\mu}, & \textbf{if} \quad \tilde{F} + \mu^{-1}Y - B < -\frac{\lambda}{\mu} \\ 0, & \textbf{otherwise} \end{cases} \tag{3.25}$$

This piecewise in the former equation is equivalent to the soft thresholding operator in Eq. 3.15. Hence, the closed-form solution of O is:

$$O = \mathcal{S}_{\frac{\lambda}{\mu}}(\tilde{F} + \mu^{-1}Y - B) \tag{3.26}$$

The complete UT-MARO algorithm is shown in Algorithm 1.

Algorithm 1 UT-MARO Algorithm

Input: F, μ, ρ
Output: O
procedure :
    $T \leftarrow CalculateTransformationDomain(F)$
    $\tilde{F} \leftarrow Alignment(F, T)$
    while not converge **do**
        $(U, \Sigma, V) = svd(\tilde{F} + \mu^{-1}Y_k - O_k)$
        $B_{k+1} = U\mathcal{S}_{\mu^{-1}}[\Sigma]V^T$
        $O_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}[\tilde{F} + Y_k - B_{k+1}]$
        $Y_{k+1} = Y_k + \mu_k(\tilde{F} - B_{k+1} - O_{k+1})$
        $\mu_{k+1} = \rho\mu_k$
    end while
end procedure

3.4.3 Method Analysis

UT-MARO reduces the computational loads in calculating the transformation via unscented transformation. However, if the features are located in the moving objects or concentrated in a specific part of the frame, the accuracy of the transformation domain will decrease.

3.5 N-MARO

The UT-MARO's disadvantages are the motive of developing N-MARO method. N-MARO method uses a variation of the Newton method (the weighted Newton method [68]) to provide more accurate results and faster convergence than the typical Newton method. Moreover, N-MARO integrates T into the problem 3.3 to reach the optimal solution when all B, O, and T are optimal. N-MARO problem formulation is as follows:

$$\min_{B,O} \ \|B\|_* + \lambda\|O\|_1 \quad s.t. \ \ F \bullet T = B + O \tag{3.27}$$

The following subsection shows the derivations of the closed-form solution of B, O, and T

3.5.1 Weighted Newton Method for Frame Alignment

The non-linearity of $F \bullet T$, due to the complicated dependence of $F \bullet T$ on T, represents the main difficulty to calculate T directly. The classical method to overcome this difficulty is to adopt Newton method (a.k.a Newton–Raphson method [91]). The basic idea of the Newton method is to approximate a non-linear function via repeated linearization. Initially, a non-linear function is approximated around an initial guess that is supposed to be close to the true root of this function, next, the resulting value

from the former approximation is used in the next approximation, and so on. Consider a system of non-linear equations $G(x) = 0$; then, this system can be expanded using Taylor series as:

$$G(x + \triangle x) \approx G(x) + G^{'}(x)\triangle x \tag{3.28}$$

where $G^{'}(x)$ is the derivative of $G(x)$; $\triangle x$ is the Newton step that approximates the root value of the system in each iteration, as shown in Algorithm 2.

Algorithm 2 Newton method for estimating non-linear systems

1: **Input:** x_0

2: **Output:** $G(x)$

3: **while** not converge **do**

4:

$$\triangle x = -\frac{G(x)}{G^{'}(x)} \tag{3.29}$$

5: $x \leftarrow x + \triangle x^*$

6: **end while**

Similarly, $F \bullet T$ is expanded using Taylor series:

$$F \bullet (T + \triangle t) \approx F \bullet T + J\triangle t \tag{3.30}$$

where J denotes the Jacobian ($J = \frac{\partial}{\partial T}(F \bullet T)$). Then, inspired by the weighted Newton method in [54], T is iteratively updated as:

$$T \leftarrow T + (\triangle t^* \times \zeta) \tag{3.31}$$

Where ζ is a scalar weighting value; $\triangle t^*$ is the optimal Newton step that is obtained

by solving the following sub-problem:

$$\underset{B,O,\triangle t}{min} \ \|B\|_* + \lambda\|O\|_1 \quad s.t. \ F \bullet T + J\triangle t = B + O \tag{3.32}$$

3.5.2 Video Decomposition

N-MARO uses IALM to decompose F into B and O. IALM reformulates Eq. 3.32 as follows:

$$\mathcal{L}(B, O, \triangle t) = \|B\|_* + \lambda\|O\|_1 + \langle Y, F \bullet T + J\triangle t - B - O\rangle + \frac{\mu}{2}\|F \bullet T + J\triangle t - B - O\|_F^2 \tag{3.33}$$

Then, the alternating strategy is used to minimize Eq. 3.33 iteratively with respect to its components, i.e. B, O, and $\triangle t$:

$$B_{k+1} = arg\min_{B} \mathcal{L}(B, O_k, \triangle t_k) \tag{3.34}$$

$$O_{k+1} = arg\min_{O} \mathcal{L}(B_k, O, \triangle t_k) \tag{3.35}$$

$$\triangle t_{k+1} = arg\min_{T} \mathcal{L}(B_k, O_k, \triangle t) \tag{3.36}$$

In the following, the closed-form solutions of B, O, and $\triangle t$ are derived from Eq. 3.34, Eq. 3.35, and Eq. 3.36. For simplicity, the subscript k in these equations is dropped when deriving the closed-form solutions.

Closed-form Solution of B

From Eq. 3.33, Eq. 3.34 is written as:

$$B = arg\min_{B} \ \|B\|_* + \langle Y, -B\rangle + \frac{\mu}{2}\|F \bullet T + \sum_{i=1}^{m} J_i\triangle t - B - O\|_F^2 \tag{3.37}$$

Since, $\|x\|_f^2 = \langle X, X \rangle$, then replace the Frobenius norm with an inner product, expand the inner product and separate B:

$$B = \underset{B}{arg\min} \|B\|_* + \langle Y, -B \rangle + \frac{\mu}{2}\|B\|_F^2 - 2\langle F \bullet T + \sum_{i=1}^{m} J_i \triangle t - O, B \rangle \tag{3.38}$$

The former equation can be simplified as:

$$B = \underset{B}{arg\min} \|B\|_* + \frac{\mu}{2}\|B\|_f^2 - 2\langle F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1}Y - O, B \rangle \tag{3.39}$$

Since, $\|B\|_f^2 = \langle B, B \rangle$, then $\langle F \bullet T + \mu^{-1}Y - O, B \rangle$ can be combined with $\|B\|_f^2$ into a single Frobenius norm:

$$B = \underset{B}{arg\min} \|B\|_* + \frac{\mu}{2}\|B - F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1}Y - O\|_F^2 \tag{3.40}$$

By applying the single value thresholding algorithm [89], the closed-form solution is

$$B = U\mathcal{S}_{\mu^{-1}}(\Sigma)V^T \tag{3.41}$$

where U, Σ and V are SVD of the term $(F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1}Y - O)$.

Closed-form Solution of O

From Eq. 3.33, Eq. 3.35 is written as:

$$O = \underset{O}{arg\min} \frac{\lambda}{\mu}\|O\|_1 + \frac{1}{2}\|O - (F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1}Y - B)\|_f^2 \tag{3.42}$$

then, obtain the partial derivative of O. However, $\|O\|_1$ function is non-differentiable. Thus, subdifferential ∂ of Eq. 3.42 is obtained and set to zero [90], as follows:

$$0 = \frac{\lambda}{\mu} sign(O) + O - (F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1} Y - B) \tag{3.43}$$

rearrange the terms

$$O = \frac{\lambda}{\mu} sign(O) + F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1} Y - B \tag{3.44}$$

and this can be expressed as:

$$O = \begin{cases} F \bullet T + J \triangle t + \mu^{-1} Y - B - \frac{\lambda}{\mu}, & \textbf{if} \quad F \bullet T + J \triangle t + \mu^{-1} Y - B > \frac{\lambda}{\mu} \\ F \bullet T + J \triangle t + \mu^{-1} Y - B + \frac{\lambda}{\mu}, & \textbf{if} \quad F \bullet T + J \triangle t + \mu^{-1} Y - B < -\frac{\lambda}{\mu} \\ 0, & \textbf{otherwise} \end{cases} \tag{3.45}$$

This piecewise in the former equation is equivalent to the soft thresholding operator in Eq. 3.21. Hence, the closed-form solution of O is:

$$O = \mathcal{S}_{\frac{\lambda}{\mu}}(F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1} Y - B) \tag{3.46}$$

Closed-form Solution of $\triangle t$

From Eq. 3.33, Eq. 3.36 is written as:

$$\triangle t = \underset{\triangle t}{arg\,\min} \|F \bullet T + \sum_{i=1}^{m} J_i \triangle t + \mu^{-1} Y - B - O\|_F^2 \tag{3.47}$$

Then, obtain partial derivative of $\triangle t$:

$$0 = F \bullet T + J\triangle t + \mu^{-1}Y - B - O \tag{3.48}$$

Rearrange the terms:

$$\triangle t = J^{\dagger}(B + O - F \bullet T - \mu^{-1}Y) \tag{3.49}$$

where $J^{\dagger}$ is the pseudo inverse of the Jacobian matrix J.

The complete algorithm of N-MARO is depicted in Algorithm 3

Algorithm 3 N-MARO Algorithm

Input: F, μ_0, ρ_0, T_0

Output: O

procedure :

  while not converge **do**

    $F \bullet T \leftarrow updateAlignment(F, T)$

    $J \leftarrow CalculateJacobin(F \bullet T)$

    while not converge **do**

      $(U, \Sigma, V) = svd(F \bullet T + J\triangle t + \mu^{-1} Y_k - O_k)$

      $B_{k+1} = U\mathcal{S}_{\mu^{-1}}[\Sigma]V^T$

      $O_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}[F \bullet T + J\triangle t + Y_k - B_{k+1}]$

      $\triangle t_{k+1} = J_k^{\dagger}(O_k + B_k - F \bullet T - \mu_k^{-1} Y_k)$

      $Y_{k+1} = Y_k + \mu_k(F \bullet T + J\triangle t - B_{k+1} - O_{k+1})$

      $\mu_{k+1} = \rho\mu_k$

    end while

    $T \leftarrow T + \triangle t^*$

  end while

end procedure

3.5.3 Method Analysis

As shown in the results section, N-MARO achieves better results than UT-MARO. Yet, N-MARO falls into problems. The first problem is the significant computational load due to using the weighted Newton method. The second problem is that B and O in N-MARO algorithm may converge faster than T because the convergence rates of IALM and the weighted Newton method are different.

3.6 I-MARO

I-MARO was proposed to solve the problems with N-MARO via the following twofold approach. First, I-MARO uses inexact Newton method to calculate T which reduces the computational load radically and enhances the accurate calculation of T. Second, I-MARO proposes a backtracking behavior to ensure that the moving objects are detected only if T reaches its optimal value. The flow chart of the I-MARO method is shown in Figure 3.2.

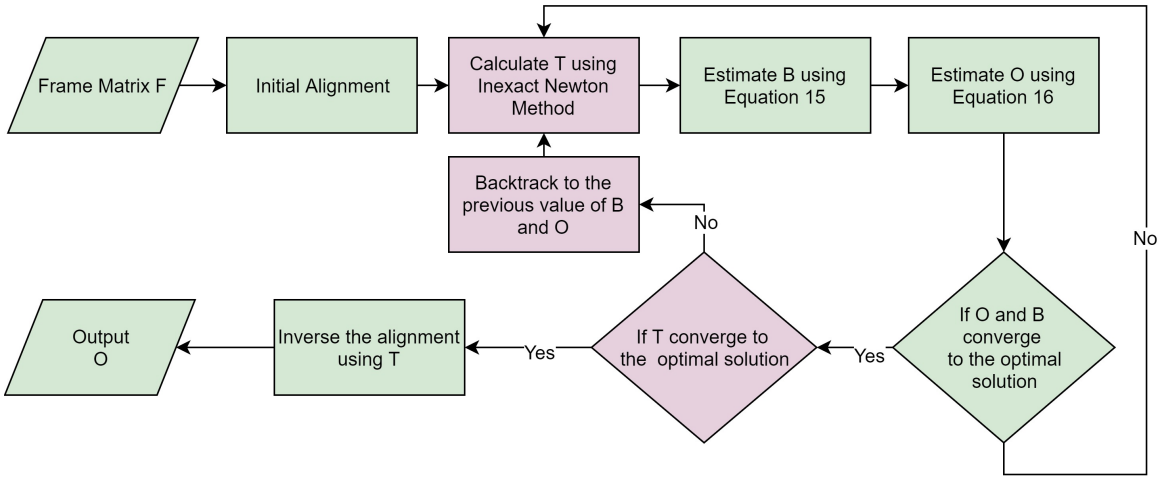


Figure 3.2: I-MARO Flowchart

3.6.1 Inexact Newton Method For Frame Alignment

Since calculating T based on feature points matching is risky, due to the reasons mentioned above, we decided to estimate T by linearising $F \bullet T$. Yet, the linearisation using the Newton method (e.g. RASL-IALM [65]), or weighted Newton method (e.g. N-MARO and DECOLOR [54]), is time consuming if the initial value T_0 is far from the solution [92]. Moreover, these Newton methods have a trade-off between the accuracy and the amount of work per iteration [93], i.e. how many iterations of the inner-loop of Algorithm 3 is required to solve the sub-problem in Eq. 3.32 to reach a certain level

of accuracy. Usually, Newton-based methods (e.g. RASL-IALM, N-MARO, etc.) set an arbitrary number of iterations to solve the sub-problem to limit the computational loads. However, this number of iterations cannot guarantee the same level of accuracy for all input datasets, because the accuracy of the calculated transformation domain T fluctuates according to camera shakiness level of each dataset.

Hence, the inexact Newton method is a better alternative because it provides an accurate and fast approximation of non-linear systems. Instead of the exact solution of the sub-problem in Eq. 3.29 of obtaining the Newton step $\triangle t^*$, the inexact Newton method adopts searching directions to generate the sequence of Newton steps without solving this sub-problem exactly. The searching directions have to satisfy an inexact Newton condition which is controlled by a forcing parameter. For a better understanding of the inexact Newton method, consider a system of non-linear equations $G(x) = 0$. The inexact Newton method calculates a sequence of Newton steps $\{\triangle x\}$ based on the searching directions that satisfy the condition in Eq. 3.50:

$$\|G(x) + G^{'}(x)\triangle x\| \leq \eta G(x) \tag{3.50}$$

where $\eta \in [0, 1)$ is the forcing parameter. Hence, instead of solving Eq. 3.29 exactly to obtain $\triangle x^*$, the forcing parameter η can be used to obtain $\triangle x$ to compute x directly, as shown in Algorithm 4:

Algorithm 4 Inexact Newton method for estimating non-linear systems

1: **while** not converge **do**
2: Find a vector $\triangle x_k$ that satisfies:
3: $\|G(x) + G^{'}(x)\triangle x\| \leq \eta G(x)$
4: $x \leftarrow x + \triangle x$
5: **end while**

The convergence of the inexact Newton method is proved in [66] and [94].

Similar to Eq. 3.50, the subjective function in Eq. 3.27 can be written using the inexact Newton method condition [66] as:

$$F \bullet T - (B + O) + J\triangle t \leq \eta(F \bullet T - (B + O)) \tag{3.51}$$

Consequently, the close-form solution to calculating the Newton step $\triangle t$ is derived as follows: Rearrange Eq. 3.51 as:

$$(1 - \eta)(F \bullet T - B - O) + J\triangle t = 0 \tag{3.52}$$

$$\triangle t = \underset{\triangle t}{arg\min}\ \langle Y, J\triangle t\rangle + \frac{\mu}{2}(1 - \eta)\|F \bullet T + J\triangle t - B - O\|_f^2 \tag{3.53}$$

Replace the Frobenious norm with an inner product, expand the inner product, and separate $J\triangle t$ as below:

$$\triangle t = \underset{\triangle t}{arg\min}\ \langle Y, J\triangle t\rangle + \frac{\mu(1 - \eta)}{2}\{\langle J\triangle t, J\triangle t\rangle + 2\langle F \bullet T - B - O, J\triangle t\rangle\} \tag{3.54}$$

$$\triangle t = \underset{\triangle t}{arg\min}\ \frac{\mu}{2}\|F \bullet T + J\triangle t + (\mu(1 - \eta))^{-1}Y - B - O\|_f^2 \tag{3.55}$$

Obtain the partial derivative for $\triangle t$

$$0 = \frac{\mu(1 - \eta)}{2}\partial(\|J\triangle t + (F \bullet T + (\mu(1 - \eta))^{-1}Y - B - O)\|_f^2) \tag{3.56}$$

Rearranging the terms, then the closed-form solution of $\triangle t$ is

$$\triangle t = J^{\dagger}(B + O - F \bullet T - (\mu(1 - \eta))^{-1}Y) \tag{3.57}$$

Finally, the transformation domain T is updated iteratively as :

$$T \leftarrow T + \triangle t \tag{3.58}$$

3.6.2 Video Decomposition

I-MARO integrates a backtracking behaviour into IALM to ensure that the optimal values of B and O are obtained only if the frames are optimally aligned, in this way, the incidence of false detections are reduced. The backtracking behaviour forces the following three conditions to be true upon convergence:

1. $d(\triangle t_{k+1}, \triangle t_k) \approx 0$
2. $|\triangle t| \approx 0$
3. $\frac{\|F \bullet T - B - O\|_f^2}{\|F \bullet T\|_f^2} \approx 0$

where $|.|$ is the absolute value; $d(,)$ denotes the Euclidean distance. According to these conditions, three convergence scenarios are considered:

- *All three conditions occur at the same time:* this scenario means the solution of the optimization problem is obtained when B, O and T converge to the optimal values concurrently.
- *First and second conditions occur before the third condition:* this scenario refers to T converging to the optimal value, but B and O are still not optimal. In this scenario, I-MARO keeps optimizing B and O with the optimal value of T until B and O converge to their optimal values.
- *Third condition occurs before the first two conditions:* This scenario implies that I-MARO reaches the optimal values of B and O before obtaining the optimal

value of T. In other words, the frames are not yet optimally aligned when the moving objects are detected. Thus, I-MARO backtracks to values of B and O at iteration $k-\kappa$ (κ is positive integer) and then keeps solving the optimization problem until one of the first two scenarios occurs.

It is worth mentioning that the backtracking behaviour does not affect the convergence proof of the original IALM; it is similar to the proved convergence in [65]. But, instead of setting B and O to zeros at each new $F \bullet T$, I-MARO uses the values of iteration number $k-\kappa$ when the backtracking is required. Algorithm 5 describes the complete I-MARO algorithm.

Algorithm 5 I-MARO Algorithm

Input: F, μ_0, ρ_0, T_0
Output: O
procedure :
  while True **do**
    $F \bullet T_{k+1} = UpdateAlignment(F, T_k)$
    $J_{k+1} = CalculateJacobin(F \bullet T_{k+1})$
    $(U, \Sigma, V) = svd(F \bullet T_{k+1} + \mu^{-1} Y_k - O_k)$
    $B_{k+1} = U\mathcal{S}_{\mu^{-1}}[\Sigma]V^T$
    $O_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}[F \bullet T_{k+1} + Y_k - B_{k+1}]$
    $\triangle t_{k+1} = J^{\dagger}_{k+1}(B_{k+1} + O_{k+1} - F \bullet T_{k+1} - (\mu(1-\eta))^{-1} Y_k)$
    $Y_{k+1} = Y_k + \mu_k(F \bullet T_{k+1} - B_{k+1} - O_{k+1})$
    $\mu_{k+1} = \rho\mu_k$
    $T_{k+1} = T_k + \triangle t_{k+1}$
    if $\frac{\|F \bullet T_{k+1} - B_{k+1} - O_{k+1}\|_f^2}{\|F \bullet T_{k+1}\|_f^2} < \epsilon$ **then**
      if $d(\triangle t_{k+1}, \triangle t_k) < \varepsilon$ and $|\triangle t_{k+1}| < \zeta$ **then**
        Break
      else
        $B_{k+1} = B_{k-\kappa}$
        $O_{k+1} = O_{k-\kappa}$
        $Y_{k+1} = Y_{k-\kappa}$
        $\mu_{k+1} = \mu_{k-\kappa}$
      end if
    end if
  end while
end procedure

Convergence

The backtracking behavior in I-MARO keeps the convergence the same as int he original N-MARO. The convergence proof was proposed in [65]. In N-MARO, IALM tries to obtain B and O at a new value of $F \bullet T$ in each iteration. Typically, $F \bullet T$ is updated at each iteration of the outer-loop of Algorithm 3. The same logic is applied in the backtracking behaviour of I-MARO; it gets the k^{th} previous values of B and O with better $F \bullet T$.

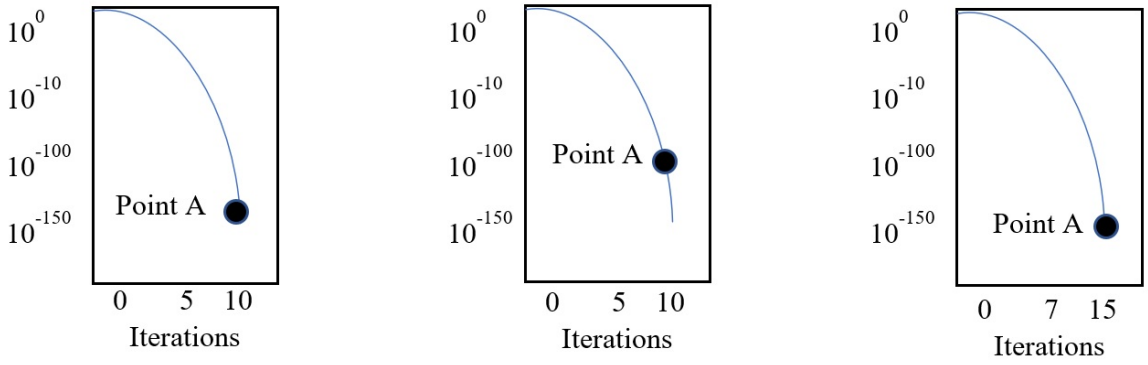


Figure 3.3: I-MARO Convergence Demonstration

Assume point A in Figure 3.3 represents the optimal value achieved using IALM. The backtracking returns point A backward to its i^{th} previous value. Then, IALM starts minimization from that value to reach the optimal value. In I-MARO case, the backtracking does not affect the convergence of IALM. Hence, IALM tries to obtain the values of B and O such that $B + O$ equals the new $F \bullet T$.

3.7 Experiment Setup

The evaluation of the proposed methods is twofold: detection accuracy evaluation and computational load evaluation. The detection accuracy evaluation is carried out on the DARPA VIVID [95], UCF aerial action [96], and VIRAT aerial [97] datasets.

Overall, ten sequences from these datasets are used for the evaluation. These sequences were carefully chosen to represent different challenges that can be found in typical aerial videos, e.g. various levels of shakiness, different environments, and diverse size of moving objects, as shown in Table 3.1. The shakiness level is computed using peak signal noise ratio (PSNR) [98] as follows:

$$PSNR = 20 \times \log_{10}(MAX) - 10 \times \log_{10} MES \tag{3.59}$$

$$MES = \frac{1}{w \times h} \sum_{\alpha=1}^{w} \sum_{\varphi=1}^{h} [f_i(\alpha, \varphi) - f_j(\alpha, \varphi)] \tag{3.60}$$

where MAX is maximum intensity value of frames f_i and f_j; w and h are the width and height of these frames. In the selected sequences, the shakiness level is between $19dB$ to $32dB$ (less dB (Decibel) means higher shakiness). The use of a variety of environments where these sequences are captured tests the detection methods under different conditions, such as, dynamic background and small displacement of the moving objects problems. For example, the dynamic background problem arises in forest environments because of the motion of the losse objects, such as tree leaves. Small displacement of the moving objects problem is obvious in urban areas or vehicle lots where the vehicles are forced to move very slowly due to traffic conditions and slow pace size of human is very small when seen from ACPs. The size of the moving objects is one of the main factors when objectively evaluating any detection method, especially in the case of aerial imagery. Thus, the moving objects' sizes in these sequences are between 15×15 pixels to 50×50 pixels.

Numerically, the detection accuracy is computed using the true positive rate (TPR) and the false positive rate (FPR), that are calculated as:

$$FPR = \frac{FP}{FP + TN} \tag{3.61}$$

Table 3.1: Datasets Characteristics Summary

Dataset	Sequences	Environment	Resolution	Object sizes	shakiness PSNR(dB)
DARPA	Egtest 1	Runway Wooden area	320 × 240	30 × 30	30
	Egtest 2			20 × 20	32
	Egtest 3			50 × 50	30
	Egtest 5			15 × 15	32
UCF	Action 1	Urban area	480 × 270	15 × 25	23
	Action 2			20 × 30	22
	Action 3			10 × 20	26
VIRAT	flight2tape1_2	Highway vehicle lots Forest	360 × 240	30 × 20	23
	flight2tape2_1			50 × 50	19
	flight2tape3_2			30 × 20	21

$$TPR = \frac{TP}{TP + FN} \tag{3.62}$$

where TP is a true detection and FP is a false detection. TN is true negative and FN is false negative. TP and FP are calculated using a region-based measure [55]. First, a binarization step is applied on the gray scale images that result from a perspective detection method being evaluated. Second, a connected components labelling algorithm is applied on the binary masks resulting from the previous step to obtain contiguous regions. Finally, these regions are counted as TP if they overlap with the ground truth, with a minimum of 25%, otherwise it is counted as FP. Since the best threshold value used in the binarization step varies from one detection method to another, the binarization is performed for different threshold values between 0 and 1 with step 0.02. Then, TPR and FPR are calculated at each threshold value, as in Eq. 3.61. The relationship between TPR and FPR is depicted using receiver operating characteristic (ROC) curves, such as Fig. 3.4, then the tip point of the corresponding ROC curve is considered the best TPR and FPR values for each of the detection method being evaluated.

The computational load evaluation is reported in two forms: execution time and complexity. The execution time is calculated on i5-4200U computer with the following

hardware specifications: CPU @1.60GHz, 2.30 GHz, x64-based processor, and 4 GB ram. The implementation of the respective detection method being evaluated is done on MATLAB. The frame resolution used in calculating the execution time is 320×240, and the execution time is computed as second(s) per frame. The complexity is evaluated per iteration as all the detection methods under evaluation in this thesis are iterative methods.

For an objective evaluation of our proposed methods, they have to be compared with relevant PCP based detection methods found in the literature–namely, RASL-IALM [65], RASL-APG [63], DECOLOR [54], and 3TD [55].

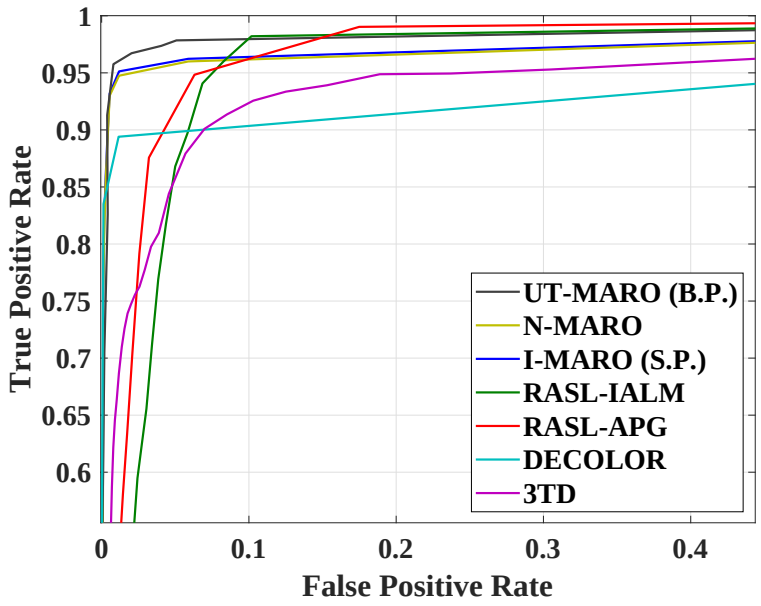


Figure 3.4: ROC curves for UT-MARO, N-MARO, I-MARO, RASL-IALM, RASL-APG, DECOLOR, and 3TD on DARPA VIVID

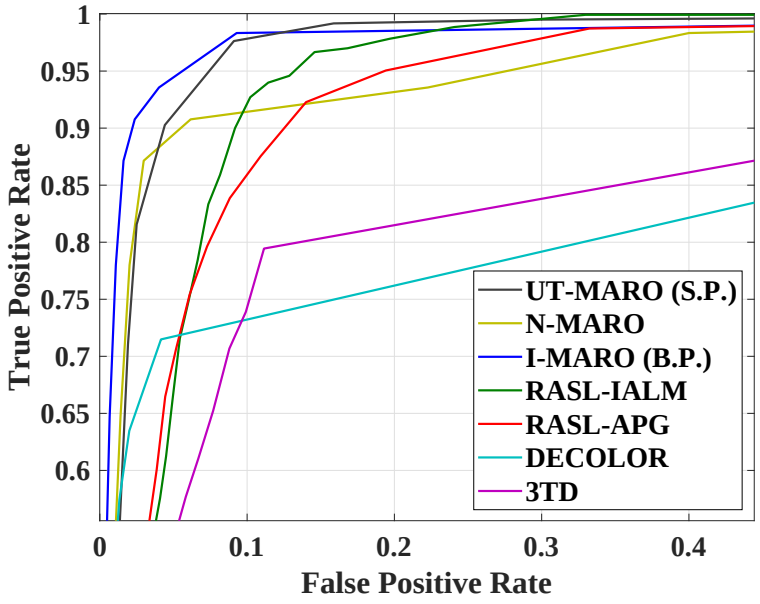


Figure 3.5: ROC curves for UT-MARO, N-MARO, I-MARO, RASL-IALM, DECOLOR, and 3TD on UCF aerial action

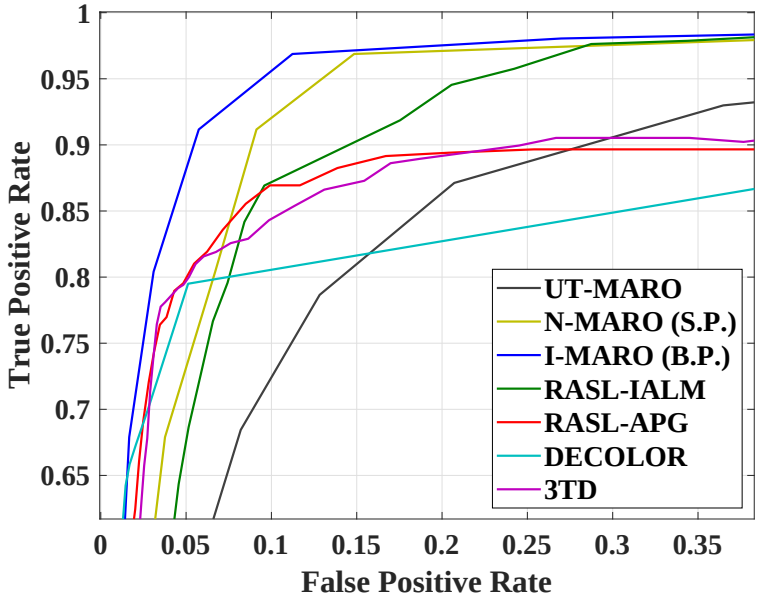


Figure 3.6: ROC curves for UT-MARO, N-MARO, I-MARO, RASL-IALM, DECOLOR, and 3TD on VIRAT

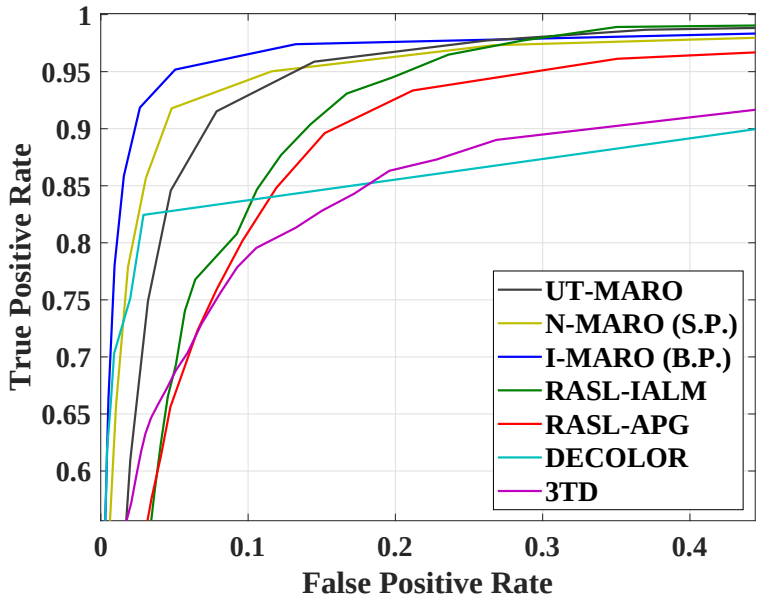


Figure 3.7: Average ROC curves for UT-MARO, N-MARO, I-MARO, RASL-IALM, DECOLOR, and 3TD

3.8 Results

3.8.1 Detection Accuracy Evaluation

As summarized in Figures 3.4, 3.5, 3.6, and 3.7, I-MARO proposes the best performance for all datasets regardless of the shakiness of the ACPs, the size of the moving object, or small movements of the background (such as those caused by wind). As shown in Tables 3.2 and 3.3, I-MARO has the highest TPR, with an average of 95.4 %; I-MARO also keeps the FPR very low, at 4.8 % on average. RASL-IALM and N-MARO have a high TPR (average 93.5 % and 93 %, respectively), but the FPR of RASL-IALM and N-MARO are high (i.e. 11 % and 8.86 % on average, respectively). In UT-MARO and RASL-APG, TPR gets lower with a higher FPR; their TPR is 90.6 % and 90.3 %, respectively; their FPR is 9.9 % and 15.3 %, respectively. DECOLOR has a low FPR, on average, 2.82 % and 6 %, respectively; however, this reduction

in their FPR is at the expense of the TPR which are 88.9 % and 79 % on average, respectively. In 3TD, the TPR is low with a high FPR; its TPR is 86.4 % and its FPR is 18.2 %.

More specifically, the smooth motion of the ACP in DARPA VIVID dataset reports high TPR with low FPR for MARO, N-MARO, and UT-MARO. However, RASL-IALM, RASL-APG and 3TD result in a high TPR and a high FPR. DECOLOR results low FPR at the expense of TPR. In the case of small background movements, (e.g. trees/branches), I-MARO reports, in general, a high TPR and a low FPR compared with the current state-of-the-art methods that report a high FPR. To support this claim, I-MARO and current state-of-the-art methods are tested using aerial imagery datasets with a dynamic background that has small movements (all the datasets have different level of shakiness caused by the movement of the aerial platform and wind). Although N-MARO and UT-MARO have a low FPR, as is evident from DARPA VIVID dataset sequence 3 (wooded area), the dynamic background and the shaky ACP in VIRAT dataset sequence 1 (Forest) and sequence 2 (Rural area) increase their FPR. Moreover, RASL-IALM, RASL-APG, and 3TD have a high FPR in the case of a dynamic background. For sample results, please refer to figures 3.8 to 3.16. For the full video sequence, please visit our website: https://phdthesisvisualresults.weebly.com/

3.8.2 Computational Load Evaluation

The best execution time with lowest complexity is achieved by UT-MARO with 1.7 seconds per frame. In the second place, I-MARO has execution time of 2.5 seconds per frame. RASL-IALM , N-MARO, and DECOLOR have execution time of 4.5, 5.8, and 6.2 seconds per frame, respectively. Despite of RASL-IALM and N-MARO use IALM as a solver for the PCP formulation, using the traditional Newton method to calculate T slows down their execution time. 3TD requires 10 seconds to process one

Table 3.2: Quantitative evaluation of I-MARO, N-MARO, and UT-MARO on DARPA VIVID, UCF aerial action and VIRAT datasets

Dataset		**UT-MARO**		**N-MARO**		**I-MARO**	
		TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Sequence 1	**97%**	1%	96%	**0.3%**	96%	**0.3%**
	Sequence 2	97%	**0.1%**	**99%**	0.7%	**99%**	0.7%
	Sequence 3	**91%**	**0.6%**	90%	1%	90%	1%
	Sequence 4	**98%**	**0.9%**	97%	**0.9%**	97%	**0.9%**
	Average	**95.7%**	**0.6%**	95.5%	0.7%	95.5%	0.7%
UCF	Sequence 1	93%	**4%**	86%	10%	**95%**	10%
	Sequence 2	81%	4%	96%	20%	**98%**	**4%**
	Sequence 3	95%	**5%**	90%	7%	**96%**	**5%**
	Average	89.6%	**4.3%**	90.6%	12.3%	**96.4%**	6.4%
VIRAT	Sequence 1	80%	40%	94%	9%	**95%**	**4%**
	Sequence 2	90%	25%	91%	23%	**92%**	**10%**
	Sequence 3	90%	10%	94%	9%	**96%**	**8%**
	Average	86.6%	25%	93%	13.6%	**94.4%**	**7.4%**
Total Average		90.6%	9.9%	93%	8.8%	**95.4%**	**4.8%**

Table 3.3: Quantitative evaluation of I-MARO versus RASL-IALM, RASL-APG, DECOLOR, and 3TD

Dataset		**I-MARO**		**RASL-IALM**		**RASL-APG**		**DECOLOR**		**3TD**	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Sequence 1	**96%**	**0.3%**	95%	6%	95%	7%	92%	1%	95%	5%
	Sequence 2	**99%**	**0.7%**	98%	6%	94%	13%	92%	1.5%	93%	12%
	Sequence 3	**90%**	**1%**	88%	7%	87%	4%	85%	1.7%	87%	20%
	Sequence 4	**97%**	**0.9%**	95%	5%	94%	2%	84%	2%	94%	8%
	Average	**95.5%**	**0.7%**	94%	6%	92.5%	6.5%	88.3%	1.6%	92.3%	11.3%
UCF	Sequence 1	**95%**	**10%**	94%	17%	92%	20%	72%	11%	85%	20%
	Sequence 2	**98%**	**4%**	95%	13%	92%	11%	71%	7%	85%	15%
	Sequence 3	**96%**	**5%**	92%	10%	91%	12%	68%	**5%**	70%	10%
	Average	**96.4%**	**6.4%**	93.7%	13.4%	91.7%	14.4%	70.4%	7.7%	80%	15%
VIRAT	Sequence 1	**95%**	**4%**	94%	9%	80%	40%	70%	5%	78%	35%
	Sequence 2	**92%**	**10%**	91%	23%	90%	25%	80%	12%	93%	30%
	Sequence 3	**96%**	**8%**	94%	9%	90%	10%	85%	9%	90%	20%
	Average	**94.4%**	**7.4%**	93%	13.7%	86.7%	25%	78.4%	8.7%	87%	28.4%
Total Average		**95.4%**	**4.8%**	93.5%	11%	90.3%	15.3%	79%	6%	86.4%	18.2%

frame. The execution time witnesses a huge hike in RASL-IALM due to the use of the APG (which is very slow) as a solver for the PCP formulation and traditional Newton method.

The complexity is calculated per iteration. It depends on the used solver for the decomposion and the calculation of T. In [67] and [99], the complexity of APG, IALM and SOFT-IMPUTE is $O(m \times n \times min(m,n))$, $O(r \times m \times n)$, and $O(\Omega + r + (M+n) \times r^2 \frac{2}{\delta})\|Z'_{\lambda_{i-1}} - Z^{\infty}_{\lambda_i}\|)$, respectively. m is the product of width and height of the frames, n is the number of frames in the frame matrix, r is the rank of the matrix; $Z^{'} = 0$ and Z^{∞} denotes the output of SOFT-IMPUTE, λ is a regularization parameter, $\Omega \subset \{1, \ldots, m\} \times \{1, \ldots n\}$ are the indices of the frame matrix. Since a simplified O notation is used in this paper, the complexity of APG, IALM and SOFT-IMPUTE is rewritten as $O(n^3)$ where n is the largest value in (m, n, r and Ω). Since I-MARO uses the inexact Newton method to calculate T, its complexity is $O(n^3)$ per iteration. In contrast, N-MARO, RASL-IALM, RASL-APG and DECOLOR use the iterative Newton method, hence their complexity is $O(n^4)$. The complexity of UT-MARO and 3TD is $O(n^3)$ as T is calculated as preprocessing using unscented transformation and dense optical flow, respectively.

Table 3.4: Runtime analysis of UT-MARO, N-MARO, I-MARO, RASL-IALM, DECOLOR, 3TD, and RASL-APG

Method	**Time (Seconds per frame)**	**Iteration Complexity**
UT-MARO	**1.7**	$O(n^3)$
N-MARO	5.8	$O(n^4)$
I-MARO	<u>2.5</u>	$O(n^3)$
RASL-IALM	4.5	$O(n^4)$
DECOLOR	6.2	$O(n^4)$
3TD	10.1	$O(n^3)$
RASL-APG	60	$O(n^4)$

3.9 Conclusion

This chapter proposes three novel PCP based methods, namely UT-MARO, N-MARO, and I-MARO. The three methods utilize IALM to solve the PCP formulation, but each one of them has a unique way to estimate the transformation domain T. UT-MARO uses unscented transformation; N-MARO uses the weighted Newton method; I-MARO uses the inexact Newton method. The results show the following facts: (1) UT-MARO reduces the computational loads, but its accuracy is not high in all cases, (2) N-MARO gives a promising accuracy, but it is very time consuming, (3) I-MARO balances between reaching high accuracy and keeping the computational loads low. Moreover, I-MARO outperforms the relative current state-of-the-art-methods (RASL-IALM, RASL-APG, DECOLOR, 3TD) in both accuracy and execution time.

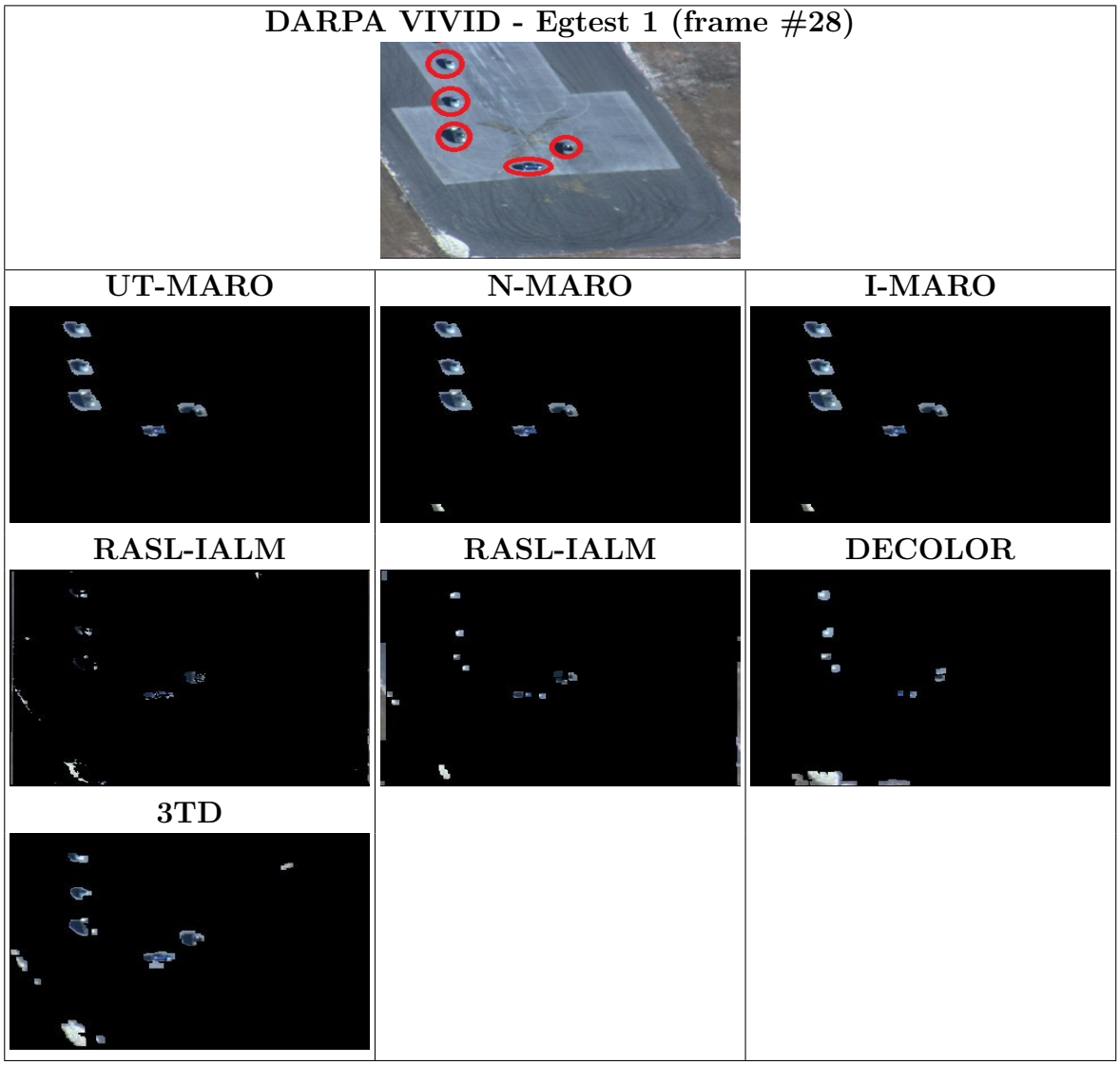


Figure 3.8: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on DARPA VIVID

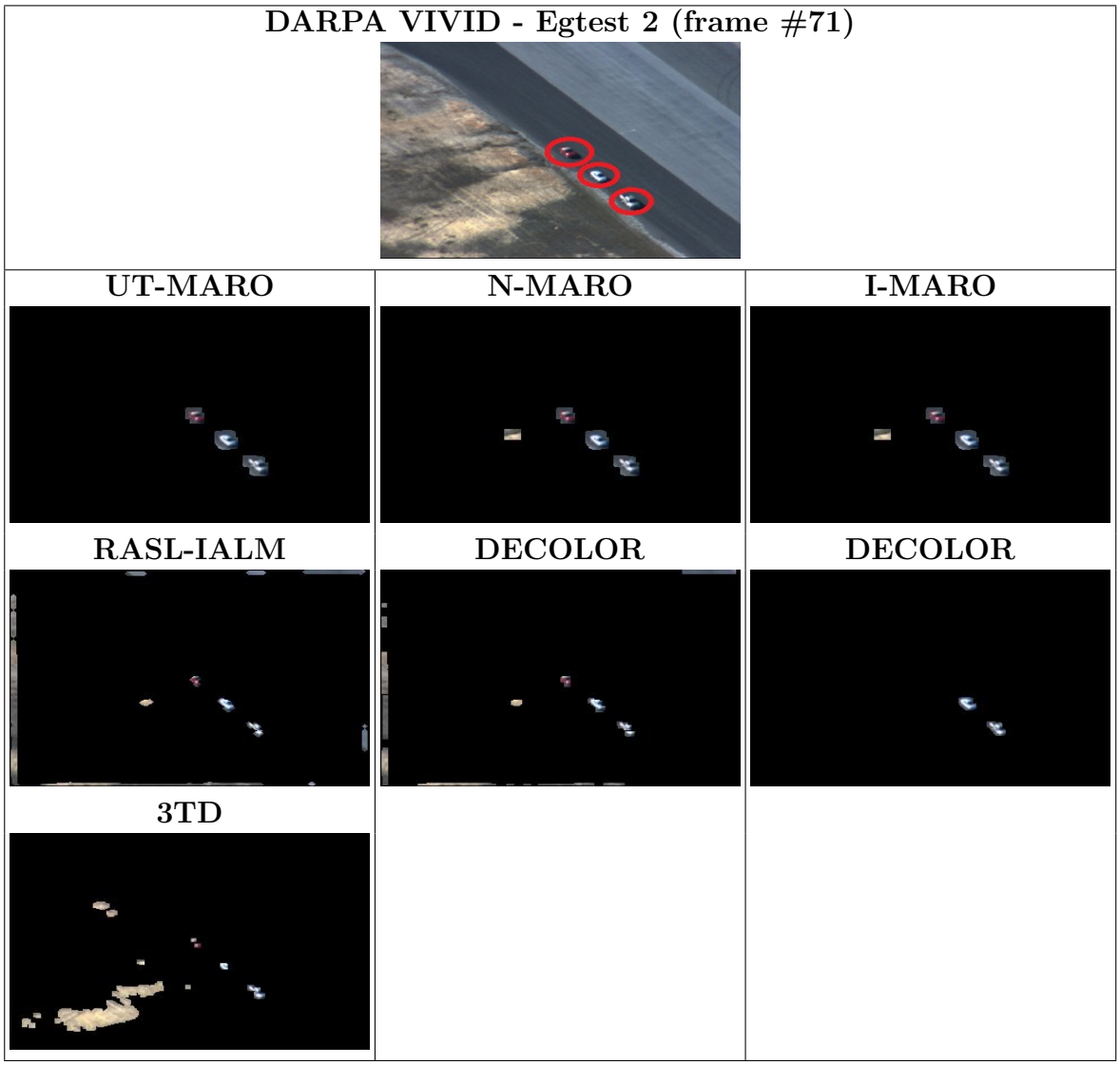


Figure 3.9: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on DARPA VIVID continued

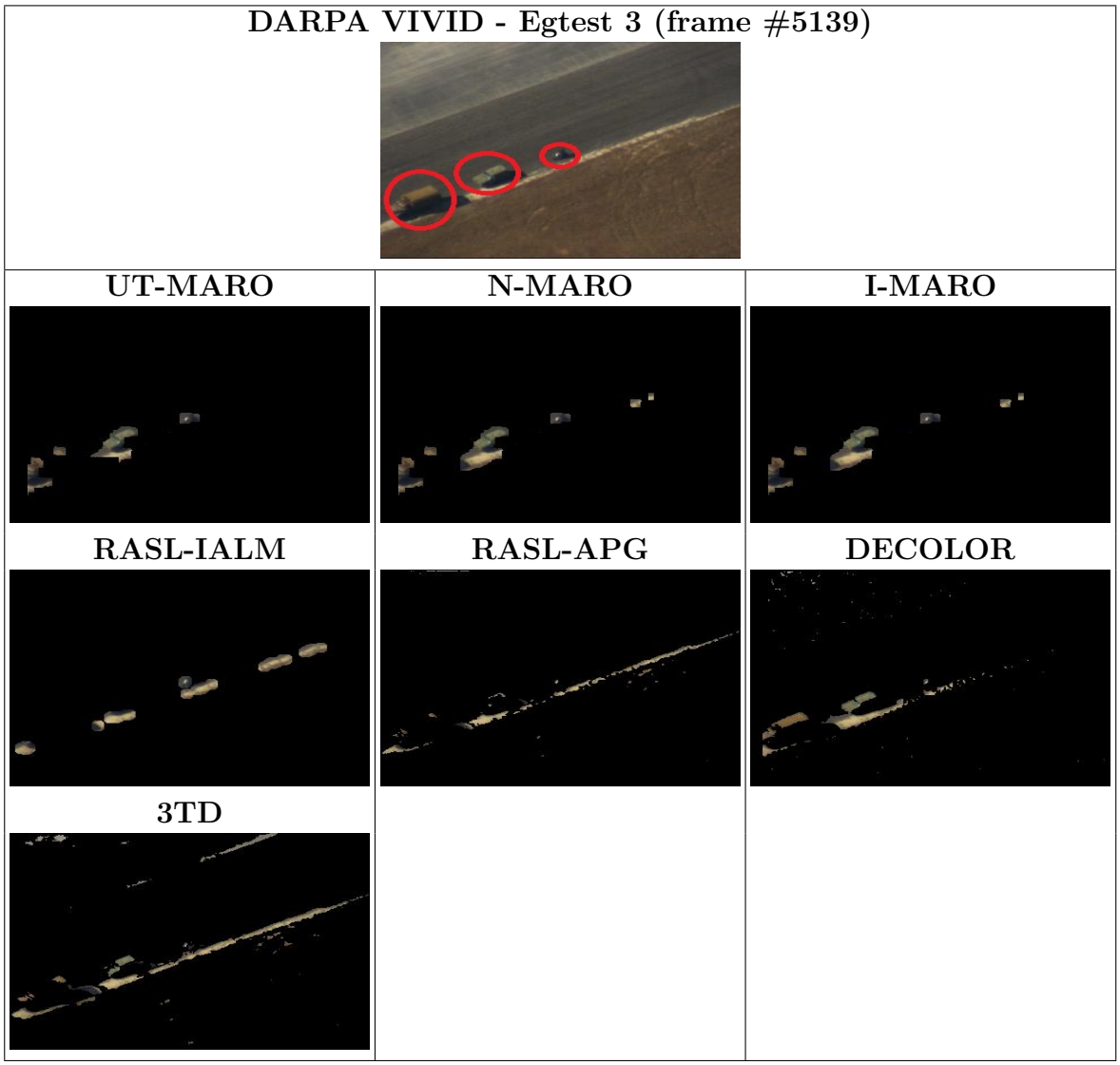


Figure 3.10: Sample results UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on DARPA VIVID continued

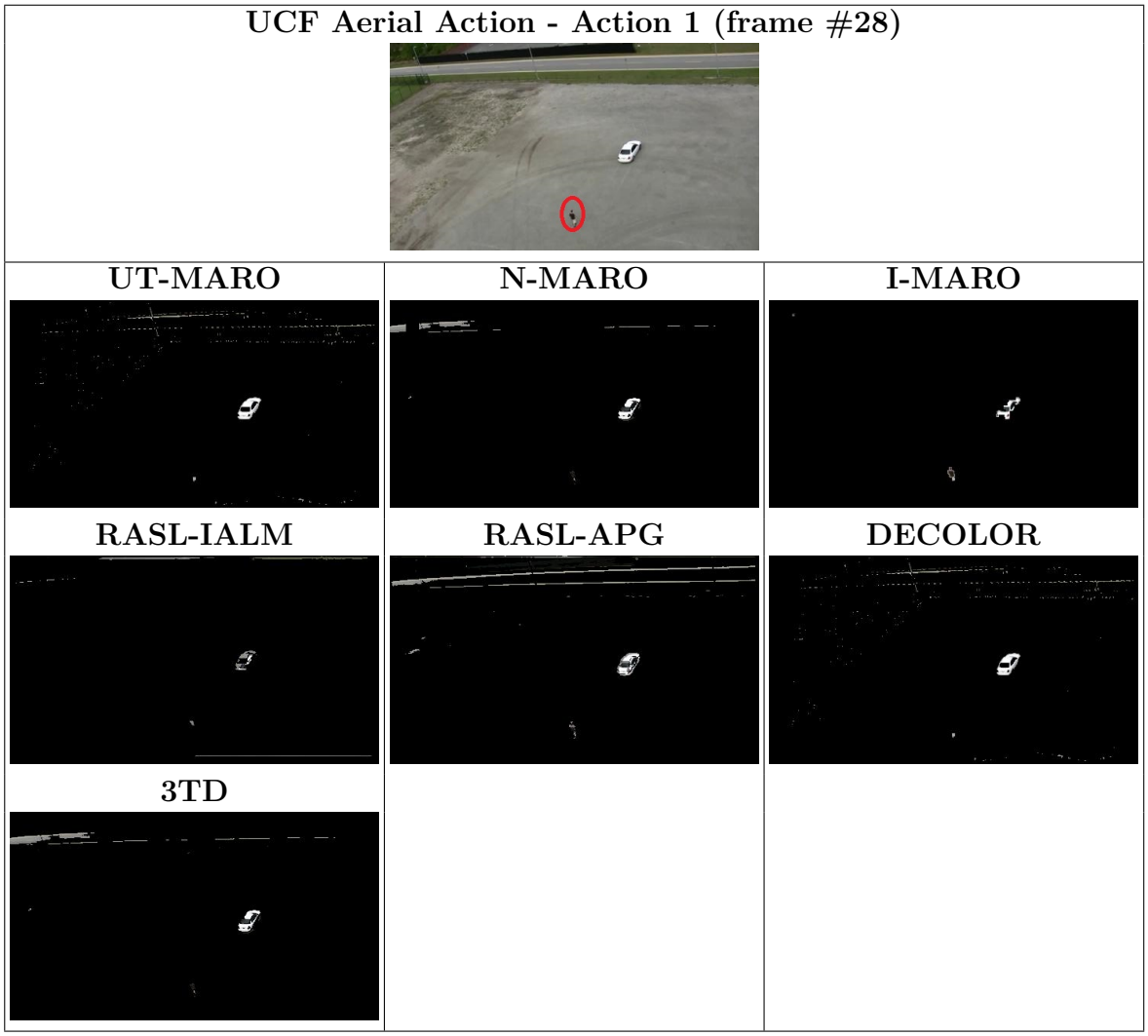


Figure 3.11: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on UCF aerial action

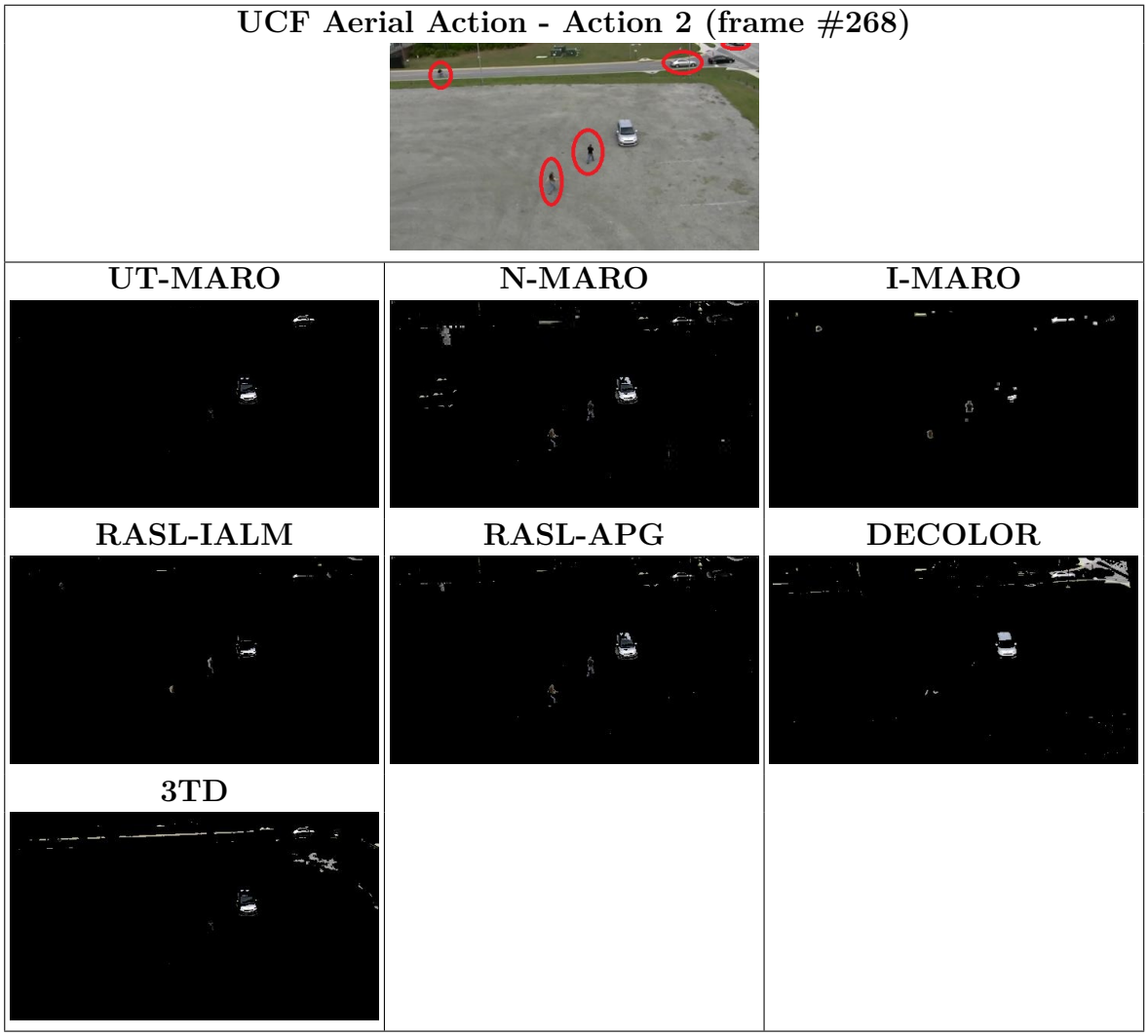


Figure 3.12: Sample results of UT-MARO, N-MARO, UT-MARO, and PCP-based detection methods on UCF aerial action continued

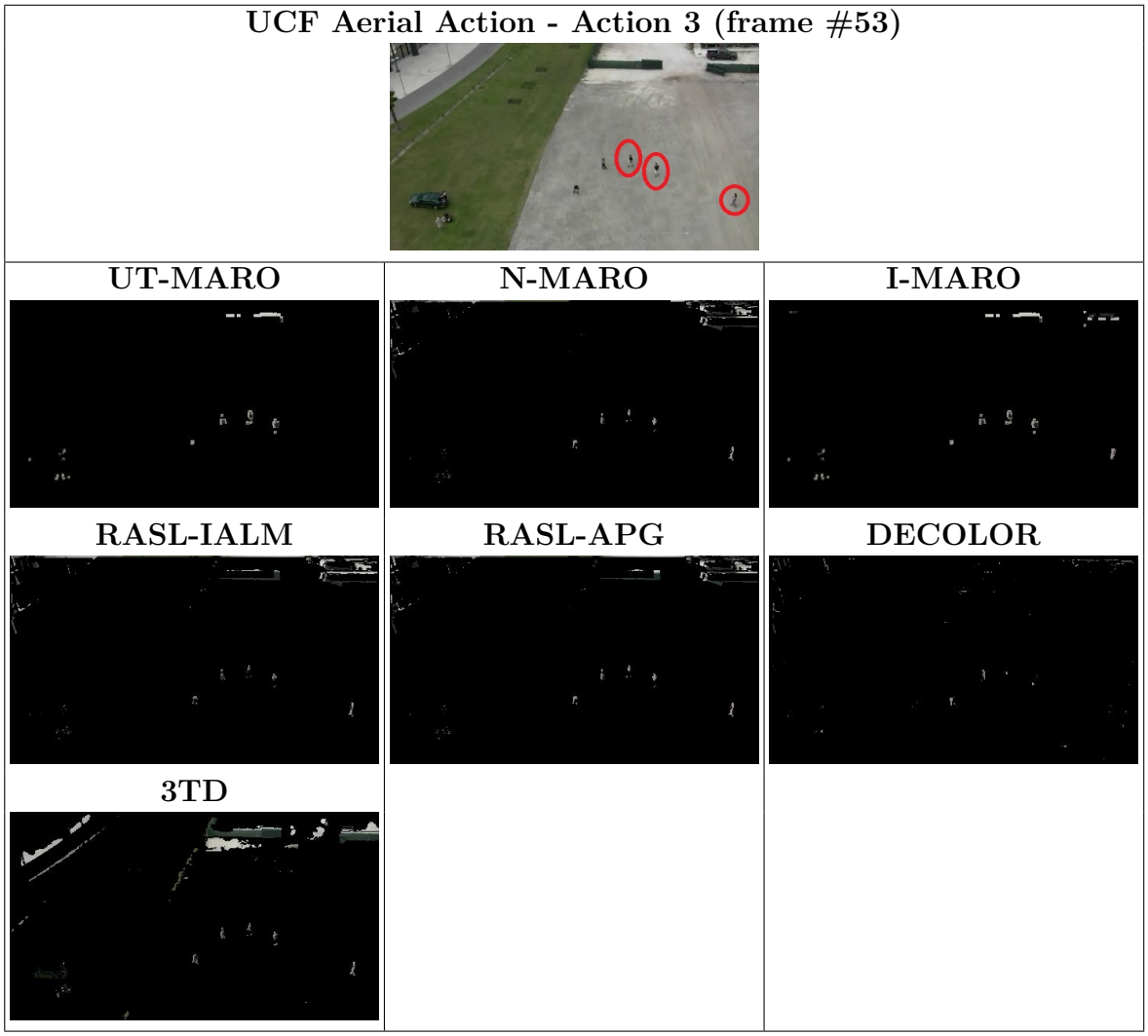


Figure 3.13: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on UCF aerial action continued

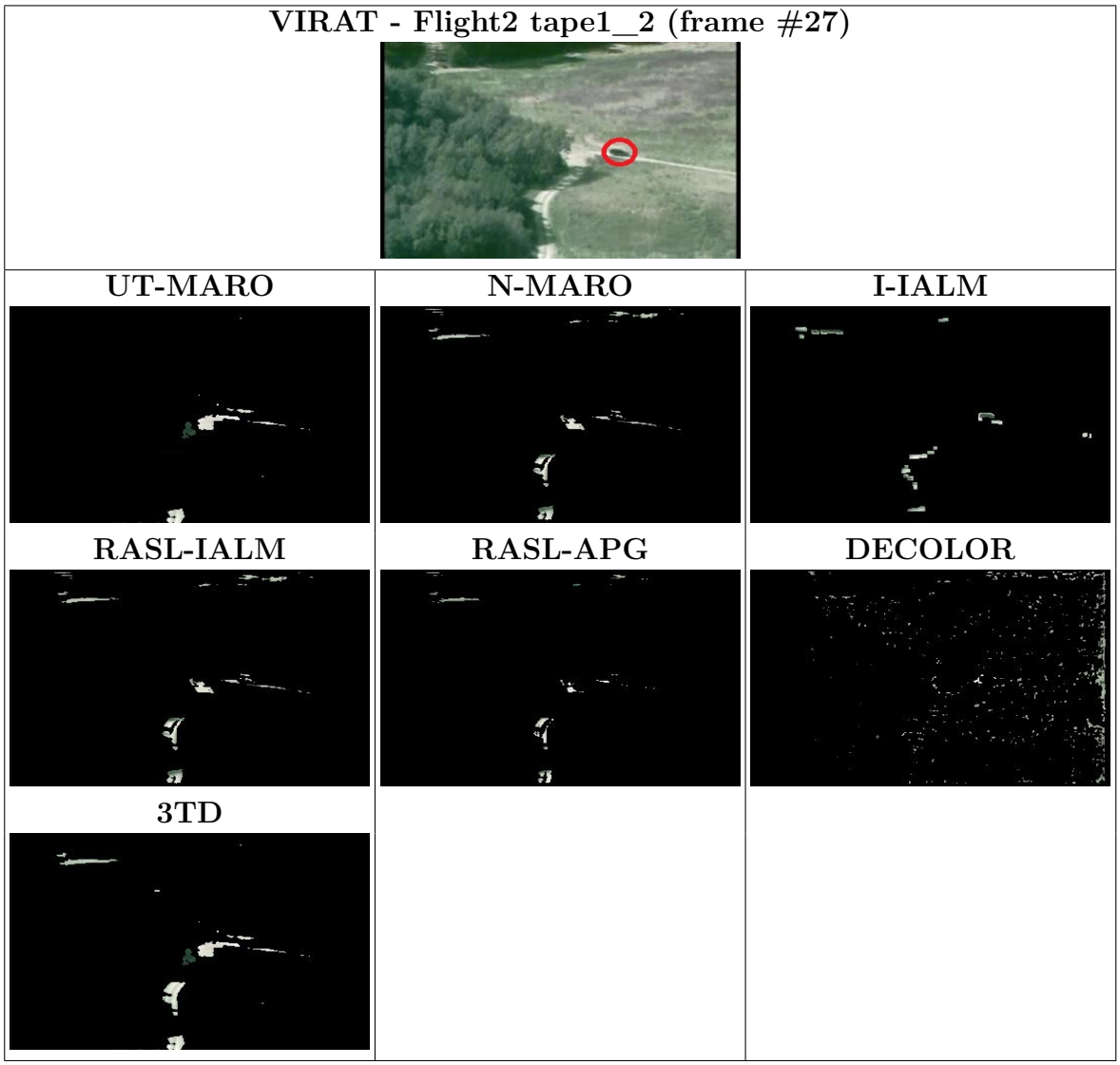


Figure 3.14: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on VIRAT

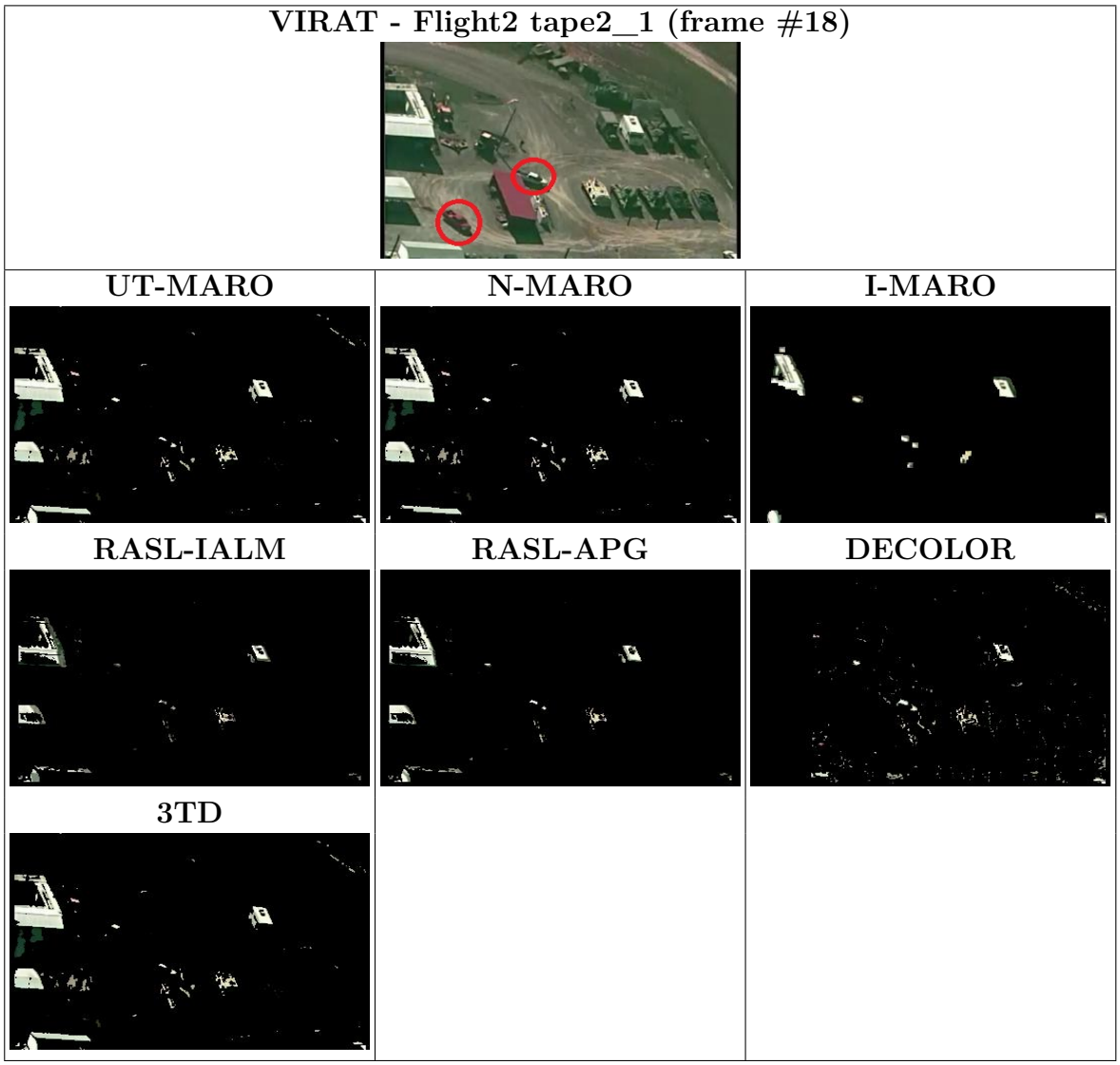


Figure 3.15: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on VIRAT continued

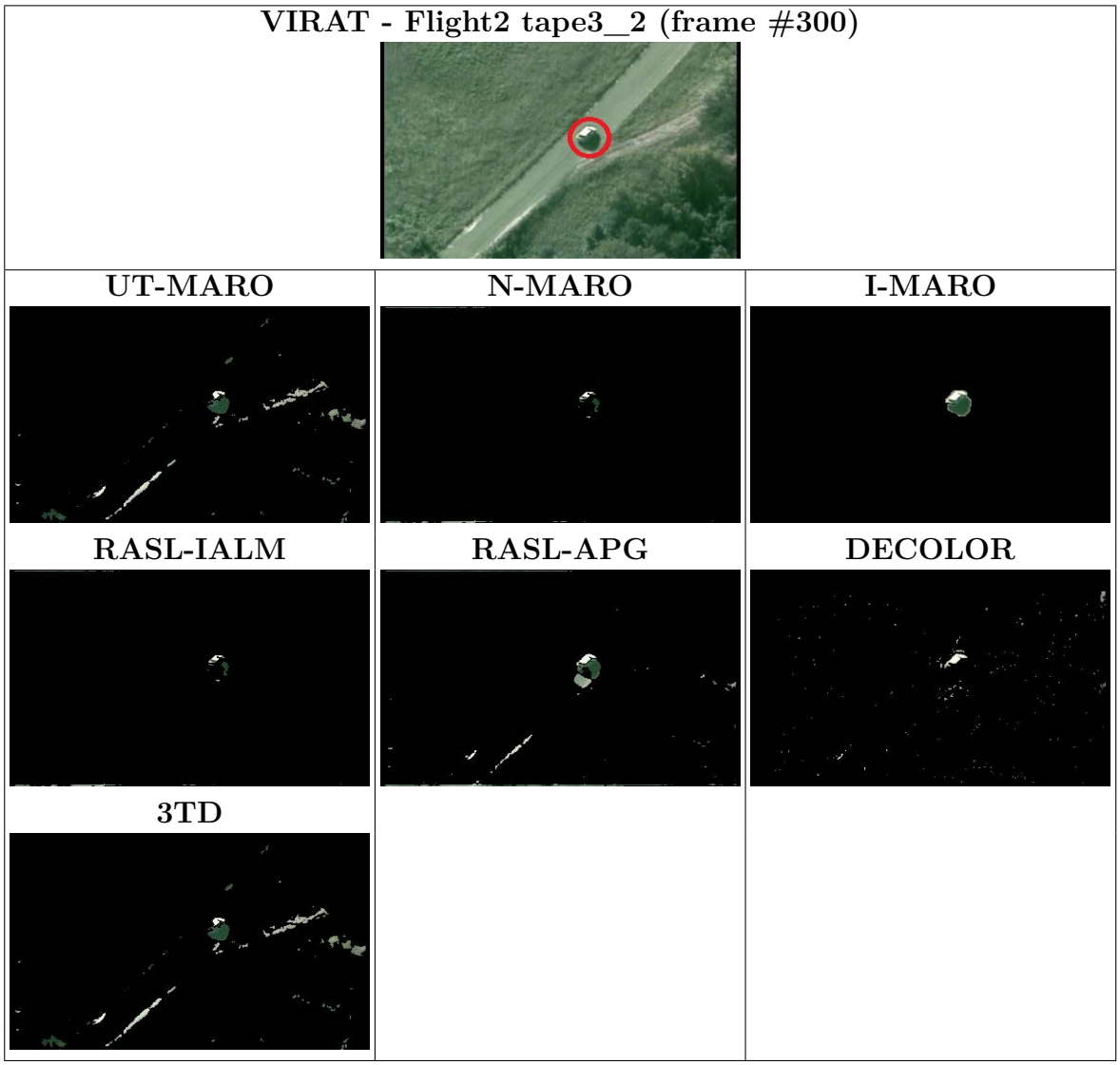


Figure 3.16: Sample results of UT-MARO, N-MARO, I-MARO, and PCP-based detection methods on VIRAT continued

Chapter 4

Principal Component Pursuit with Regularization Term

4.1 Introduction

The previous chapter shows how the detection accuracy is improved via I-MARO. Yet, FPR in all the proposed methods, including I-MARO, is considerable. For example, UT-MARO has FPR 9.9 %, N-MARO has FPR 8.8 %, and I-MARO has FPR 4.8 %. The high FPR can be justified as modelling moving objects as sparse (in Eq. 4.1) is very general and it could be applied on any background object in the scene, especially with existence of a little misalignment between frames. For example in Figure 4.1, I-MARO method detects the white car as a moving object, but this car is not moving. This is because of a small misalignment between frames that makes the car to be considered as a sparse. In conclusion, modelling moving objects as sparse cannot guarantee that the detected moving objects matrix O contains only

truly moving objects; hence, the false detection rate is considerable.

$$\min_{B,O} \; Rank(B) \quad s.t. \;\; F = B + O, \;\; \|O\|_0 \leq \varepsilon \tag{4.1}$$

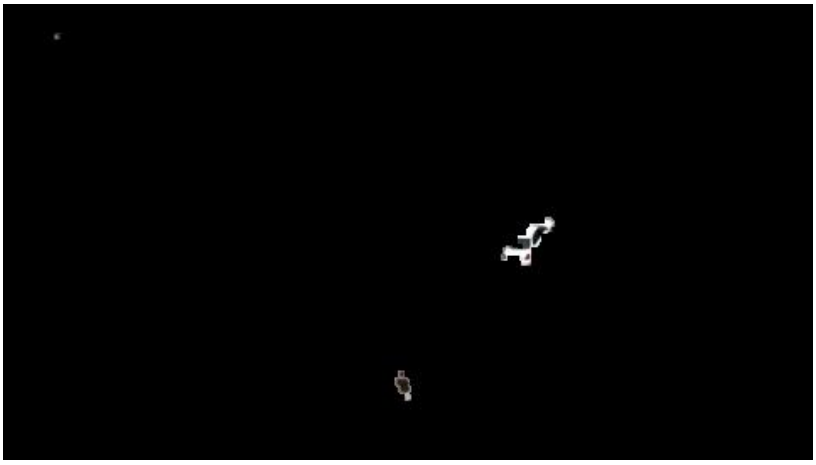


Figure 4.1: Applying I-MARO on Frame 28 of UCF Aerial Action 1

4.2 Chapter Contributions

This chapter focuses on addressing the former problem to ensure that O has only truly moving objects and significant reduction in the false detections. To this end, a regularization term is integrated into PCP formulation 4.2. The regularization term models the motion of moving objects in comparison with their surrounding static objects. Through this modelling, the regularization term determines the most likely moving object regions; hence, it limits the solution of O to true moving objects. In other words, instead of modelling moving objects as sparse, using the regularization term in PCP formulation models the moving objects as moving sparse.

In this chapter, a new PCP formulation is proposed. The work in this chapter is presented in the following journal and conference papers:

- Agwad ElTantawy and Mohamed S. Shehata. "KRMARO: Aerial Detection of Small-Size Ground Moving Objects Using Kinematic Regularization and Matrix Rank Optimization." *IEEE Transaction on Circuits and System for Video Technology* 2018. DOI: 10.1109/TCSVT.2018.2843761.

- Agwad ElTantawy, and Mohamed S. Shehata. "A novel method for segmenting moving objects in aerial imagery using matrix recovery and physical spring model." *In Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 3898-3903. IEEE, 2016.

The results in this chapter are replicated from these two publications.

4.3 The Physical Intuition of The Regularization Term

Assume you are in an airplane and looking from a side window down toward a road. You observe a car on the ground and attempt to determine whether this car is moving or not. Unfortunately, it is not an easy task due to the continuous motion of the airplane. This continuous motion makes the whole scene to be moving; hence, differentiating actual moving objects from non-moving is difficult. This problem can be resolved for objects that are static by nature, e.g. buildings, trees, etc.; however, how about mobile objects, such as cars, human, bikes, etc. These mobile objects could be in the scene but not moving. Moreover, the large distance between the airplane and the car that you are observing makes the detection even harder. This distance makes everything appear very small; hence, it will be hard to depend on spatial features of the car to figure out if it is moving or not.

The best approach to handle this former problem is to compare the location of

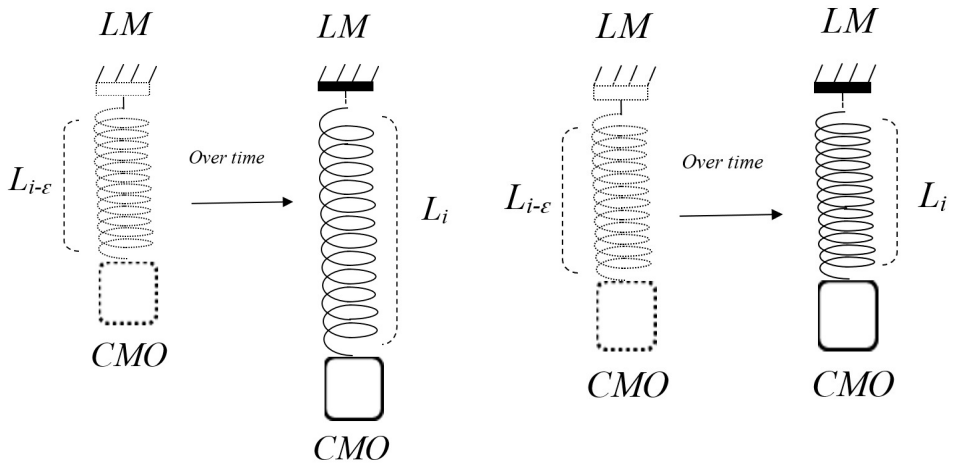
the car with the location of known static objects, e.g. trees, building, etc., over time. If the car is moving, the distance between this car and the static objects will be changed over time, i.e., the relative position will have changed and the distance will have increased or decreased. On the other hand, the distance between the car and static objects is fixed over time when the car is not moving.

Mathematically, the former process can be modelled using a mass-spring's force function to track the variation in the distances between objects that might be moving and static landmarks. This force function provides a general modelling of the motion of these objects' motion with respect to the static landmarks over time, e.g. the motion of an object towards and/or apart from the landmarks over time. Therefore, spring force function is chosen to sufficiently model the variation over time between moving objects and landmarks. After calculating the mass-spring's force function for each object, the regularization term uses the resulting forces to determine if this object is moving or not. The following sections illustrates the details of calculating the mass-spring's force function and how the regularization term works.

4.4 Spring-MARO: Spring model with MAtrix Rank Optimization

4.4.1 Problem Formulation

Spring-MARO models the moving objects as moving sparse. It detects the truly moving objects in two steps: 1) extracting candidate moving objects and 2) refining the candidate objects. In the first step, Spring-MARO apply the matrix rank optimization concepts that were presented in the previous chapter. The output of this step is the identification of candidate moving objects (CMO). In the second step, the physical



(a) True moving object (b) False detection

Figure 4.2: Transitional motion effect on compression spring over time

spring model classifies these candidate moving objects into false detections and true moving objects. Each candidate moving object is modelled as a mass suspended to a compression spring, and the spring base is attached to a manually selected landmark LM which is a part of the scene. If a CMO is a true moving object, the mass should make this spring either compressed or stretched over the time due to the motion of the object, as shown in Figure 4.2-a, hence, the force of the spring will be significant. On the other hand, the spring will be static and its force is zero when the object is not moving, as shown in Figure 4.2-b.

From the above, the detection problem can be formulated as:

$$SM(\min_{B,\check{O},T} \|B\|_* + \lambda\|\check{O}\|_1, s.t.\ F \bullet T = B + \check{O}) \tag{4.2}$$

where $\check{O} \in \mathbb{R}^{n\times m}$ contains all candidate moving objects $CMOs$ in F; SM denotes a spring model function that rejects all possible false detections that are detected in $\check{O}$. As in the previous chapter, B is the underlying background, $F \bullet T$ denotes the frames matrix aligned using the transformation domain T, $\|X\|_*$ is the nuclear norm,

and $\|X\|_1$ denotes $L_1 - norm$.

4.4.2 Extracting Candidate Moving Objects

Spring-MARO makes an initial extraction of candidate moving objects by solving the sub-problem Eq. 4.3 in the same strategy as N-MARO that was presented in the previous chapter. Typically, the weighted Newton method calculates T, while IALM solves the optimization problem via an alternative strategy with respect to B, $\check{O}$, and T.

$$\underset{B,\check{O},T}{min}\|B\|_* + \lambda\|\check{O}\|_1, s.t.\ F \bullet T = B + \check{O} \tag{4.3}$$

The closed form-solutions of B, $\check{O}$, and T are shown in Eq. 4.4, Eq. 4.6, and Eq. 4.7, respectively.

$$B = U\mathcal{S}_{\mu^{-1}}(\Sigma)V^T \tag{4.4}$$

$$[U, \Sigma, V] = F \bullet T + J\triangle t + \mu^{-1}Y - \check{O} \tag{4.5}$$

$$\check{O} = \mathcal{S}_{\frac{\lambda}{\mu}}(F \bullet T + J\triangle t + \mu^{-1}Y - B) \tag{4.6}$$

$$T \leftarrow T + \triangle t \tag{4.7}$$

where $\triangle t$ is calculated as follows:

$$\triangle t = J^{\dagger}(F \bullet T + \mu^{-1}Y - B - \check{O}) \tag{4.8}$$

$J^{\dagger}$ is the pseudo inverse of the Jacobian matrix J.

4.4.3 Spring Model for Truly Moving Objects Detection

To calculate the spring force of a candidate moving object (CMO), it has to be tracked over time, i.e. the location of this object has to be known in at least one

previous frame, as well as the landmark (LM). To this end, a correlation tracker with a searching window is used to detect the locations of $\check{O}$ and LM in frames f_i and $f_{i-\varepsilon}$. In the case of aerial imagery, matching $\check{O}$ among the frames using features, such as SURF [88], FAST [100], etc., is very difficult as these objects appear very small. Generally, small objects suffer from lack of strong features, consequently wrong matching is very frequent. Therefore, the matching in the proposed method is accomplished using gradient template matching, as shown in Figure 4.3, in which cross-normalization [101] is applied between the gradient of the CMO and the gradient of the frame $f_{i-\varepsilon}$. To avoid ambiguous matching, cross-normalization is done within a searching window of frame $f_{i-\varepsilon}$. This window is defined as an area where the moving object is expected to be in frame $f_{i-\varepsilon}$. In our experiments, the best width and height of the searching window are 100 + width and 100 + height of the CMO, respectively. However, the width and height of the searching window may be changed for other datasets based on the objects speed, flight altitude, and focal length. The location of the CMO is detected at frame f_i from the previous step, i.e. extracting candidate moving objects, while the location of LM is detected in both frames f_i and $f_{i-\varepsilon}$ using a pre-defined template.

After locating $CMOs$ and their corresponding LMs, the force function of the spring that connects them is calculated based on the difference in the distance between CMO and LM in frames f_i and $f_{i-\varepsilon}$, as follows:

$$CSF(CMO, LM) = K(L_{i-\varepsilon} - L_i) \tag{4.9}$$

$L_{i-\varepsilon}$ and L_i represent the length of a spring in the system of spring frame $f_{i-\varepsilon}$ and frame f_i, respectively; CSF denotes compression spring force. K is spring constant and it is equal to unity, in our case. Other forces that may affect the system of springs

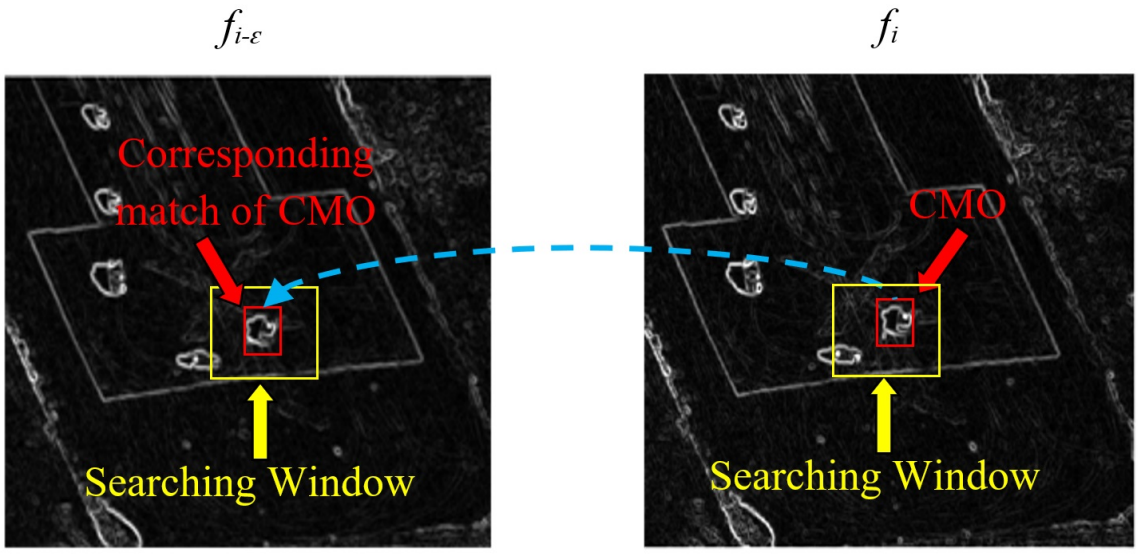


Figure 4.3: Tracking CMO in frame $f_{i-\varepsilon}$

are ignored, such as gravity force, dampening force, etc., as they are irrelevant to moving object segmentation.

To capture the motion in all directions, each CMO is attached with 8 springs in different directions as shown in Figure 4.4. Then, the total forces of these springs are used to judge whether $\check{O}$ is moving or not using the regularization term SM, as in Eq. 4.10.

$$O = SM(\check{O}) \tag{4.10}$$

where $SM(\check{O})$ is defined as

$$O = \begin{cases} 1, & TF > \varphi \\ 0, & TF < \varphi \end{cases} \tag{4.11}$$

$O \in \mathbb{R}^{n \times m}$ denotes the true moving objects; φ is an arbitrarily defined threshold value; and TF is the total forces of the system of springs that is calculated from Algorithm 6.

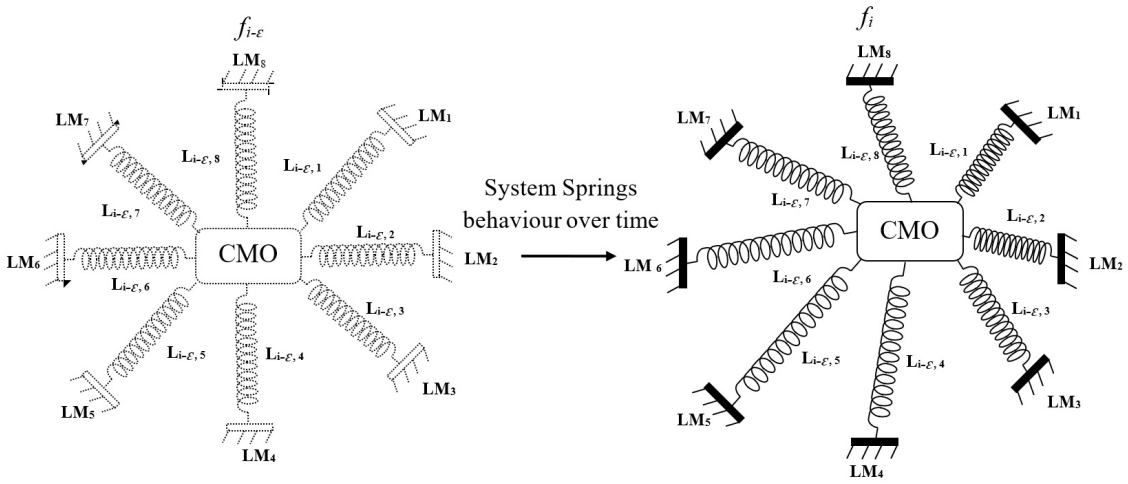


Figure 4.4: System of Spring Setup

4.4.4 Method Analysis

Modelling the moving objects as moving sparse gives promising detection accuracy, as will be shown in the results section. But Spring-MARO suffers from severe drawbacks that may limit its accuracy. First, it assumes that the template of the LMs is not changing in all the frames of a sequence, which is not the case in aerial imagery due the camera motion. Second, an error in detecting the LMs location in one frame leads to wrong results for all $CMOs$ in this frame. Third, the spring model does not consider the rotational motion of the moving objects.

4.5 KR-MARO: Kinematic Regularization and Matrix Rank Optimization

KR-MARO proposes a kinematic regularization term to the PCP formulation that is based on an advanced spring model to overcome the drawbacks of SM in Spring-MARO method. This kinematic regularization term enforces the solution O to have moving objects in regions that have unique kinematic properties. Since, the motion of

Algorithm 6 Calculate total force TF for candidate moving objects in frames f_i

1: **procedure**
2: $\check{f}_i = vec^{\dagger}(\check{O}[i])$ *// $\check{f}_i$ has 1s at CMO and 0s any where else. $vec^{\dagger} : \mathbb{R}^n \rightarrow \mathbb{R}^{w \times h}$*
3: Obtain $CMOi, j$ in f_i using connect component analysis
4: **for** `(j=1 ; j<=ComponentsCount ; j++)` **do**
5: $\upsilon = \Sigma_{w=1}^{8} CSF(CMO_{i,j}, LM_w)$
6: Update $(\check{f}_i)$ with υ at location of $CMO_{i,j}$
7: **end for**
8: $TF[i] \leftarrow vec(f_i)$ *//$vec : \mathbb{R}^{w \times h} \rightarrow \mathbb{R}^n$*
9: **end procedure**

a moving object can be translational or rotational or both, kinematic properties of a moving object include translational and rotational motion. These two types of motion are captured by compression and torsion springs, respectively. Contrary to Spring-MARO, the selection of LMs is done automatically and they are updated for each CMO at each frame. KR-MARO uses the inexact Newton method to calculate the transformation domain, and backtracking behaviour in IALM to obtain the moving objects only when the frames are optimally aligned.

4.5.1 Problem Formulation

KR-MARO integrates the kinematic regularization term into the PCP formulation as follows:

$$\underset{B,O,T}{min} \|B\|_* + \lambda \|\Gamma(O)\|_1, s.t.\ F \bullet T = B + O \tag{4.12}$$

where $\Gamma : \mathbb{R}^{r \times n} \rightarrow \mathbb{R}^{r \times n}$ is the kinematic regularization; it is a linear operator that weighs the entries of O according to their kinematic properties, where the truly moving objects are unchanged while false detections are set to zero. In fact, the integration of the kinematic regularization into the PCP formulation leads to two types of sparse:

truly moving objects and false detections. In other words, the frames matrix should be decomposed into the background (B), the truly moving objects (O), and false detections (D). Hence, KR-MARO is problem formulation is:

$$\underset{B,O,D,T}{min} \|B\|_* + \lambda\|\Gamma(O)\|_1, s.t.\ F \bullet T = B + O + D \tag{4.13}$$

4.5.2 Closed-Form Solution

KR-MARO proposes a solution of the problem formulation in Eq. 4.13 based on the inexact Newton method and IALM with backtracking behaviour. First, the formulation is approximated as follows:

$$\begin{aligned}\mathcal{L}(B,O,D,\triangle t) = \quad & \|B\|_* + \lambda\|O\|_1 + \langle Y, F \bullet T + J\triangle t - B - O - D\rangle + \\ & \tfrac{\mu}{2}\|F \bullet T + J\triangle t - B - O - D\|_F^2\end{aligned} \tag{4.14}$$

Then, the former equation is solved by minimizing with respect to its components, i.e. B, O, D, and $\triangle t$. Inspired by the derivations in the previous chapter, the closed-form solution of B, O, D, and $\triangle t$ are obtained as follows:

- The closed-form solution of B is

$$B = U\mathcal{S}_{\mu^{-1}}(\Sigma)V^T$$

  where U, Σ and V are the singular value decomposition (SVD) of the term $(F \bullet T + \mu^{-1}Y - O - D)$.

- The closed-form solution of O is:

$$O = \mathcal{S}_{\frac{\lambda}{\mu}\Gamma}(F \bullet T + \mu^{-1}Y - B - D)$$

To show how Γ controls the results in O, assume $X = F \bullet T + \mu^{-1}Y - B - D$. From the closed-form solution of O, X could fall in region 1, 2, or 3, as shown in Figures 4.5 and 4.6. If X is in Regions 1, then X is zero and its corresponding entry in O is considered as background. However, if X is in regions 2 or 3, its corresponding entry in O is a moving object. Γ controls the boundaries of regions 1, 2, and 3. When Γ is zero, then region 1 will disappear, and in this case the corresponding entry of X in O is a moving object, as shown in Figure 4.6. However, if Γ is a very large value, then X will be in region 1 which mean its corresponding entry O is a non-moving object, as shown in Figure 4.7.

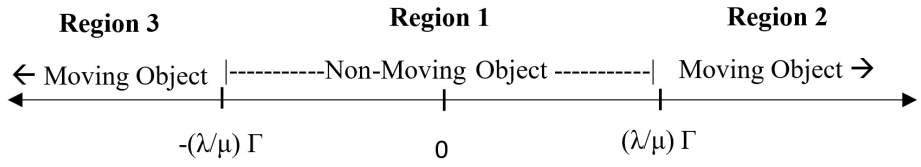


Figure 4.5: possible values of the entries in O matrix

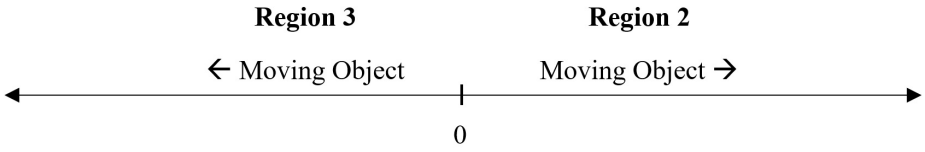


Figure 4.6: Regularization term Γ is zero:

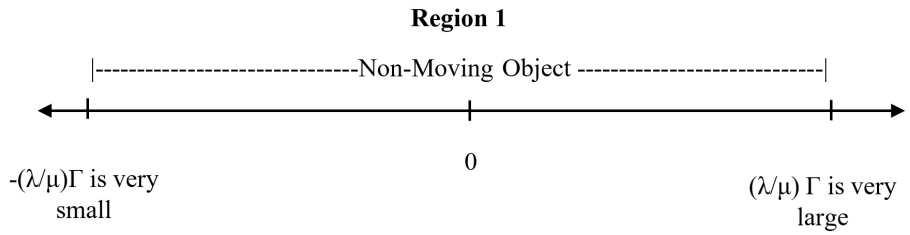


Figure 4.7: Regularization term Γ is very large

- The closed-form solution of D is derived as follows:

$$D = \underset{D}{arg\min}\ \langle Y, -D\rangle + \frac{\mu}{2}\|F \bullet T - B - O - D\|_f^2 \tag{4.15}$$

$$D = \underset{D}{arg\min}\ \frac{1}{2}\|D - (F \bullet T + \mu^{-1}Y - B - O)\|_f^2 \tag{4.16}$$

$$0 = \frac{1}{2}\partial(\|D - (F \bullet T + \mu^{-1}Y - B - O)\|_f^2) \tag{4.17}$$

$$D = F \bullet T + \mu^{-1}Y - B - O \tag{4.18}$$

- The closed-form solution of $\triangle t$ is:

$$\triangle t = B + O + D - J^{\dagger} - F \bullet T - (\mu(1-\eta))^{-1}Y \tag{4.19}$$

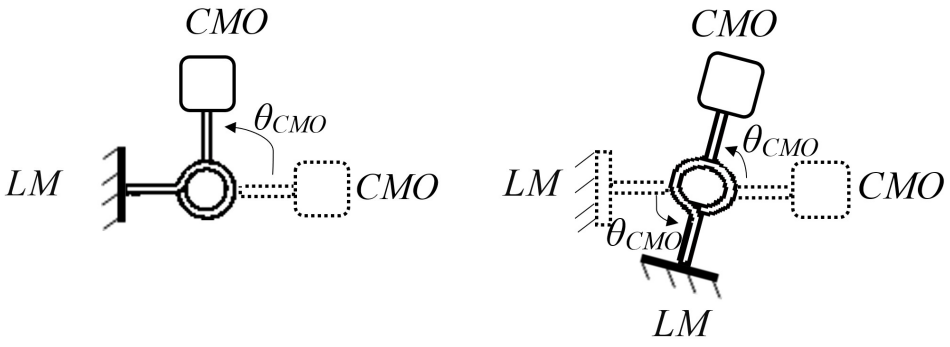
4.5.3 Kinematic Regularization Term

Modelling the Motion of CMO Using Compression and Torsion Springs

To judge the uniqueness of the kinematic properties of a CMO, it is modelled as a mass suspended from two types of springs: a compression spring and torsion spring, as shown in Figures 4.2 and 4.8. When the CMO is truly moving, at least one of the former springs has a significant force (due to the compression and/or stretching in the compression spring and/or twisting in the torsion spring). On the other hand, the total forces of these springs is zero if the object is not moving.

Calculating for Forces of Compression and Torsion Springs

Similar to Spring-MARO, the first step to calculate a spring force is to track the location of CMO and its corresponding LMs over the time using the gradient template matching (Figure 4.3). Then, the force of the compression spring is calculated as



(a) True moving object (b) False detection

Figure 4.8: Rotational motion effect on torsion spring over time

shown in Eq. 4.9. The force of torsion springs are calculated as

$$TSF(CMO, LM) = K(|\theta_{CMO} - \theta_{LM}|) \tag{4.20}$$

where TSF is the Torsion Spring Force. θ_{CMO} is the angle between the locations of a O_{CMO} in f_i and f_{i+m}, respectively, while θ_{LM} is the angle between the locations of LM of this O_{CMO} in f_i and f_{i+m}, respectively. These angles are calculated using the difference in y divided over the difference in x between two frames as follows:

$$\theta_{CMO} = tan^{-1}\left(\frac{\triangle y_{CMO}}{\triangle x_{CMO}}\right),\ \theta_{LM} = tan^{-1}\left(\frac{\triangle y_{LM}}{\triangle x_{LM}}\right) \tag{4.21}$$

where $\triangle y_{CMO}$ and $\triangle x_{CMO}$ are the differences of the CMO locations on Y-axis and X-axis, respectively, in f_i and f_{i+m}; $\triangle y_{LM}$ and $\triangle x_{LM}$ are the differences of the LM locations on Y-axis and X-axis, respectively, in f_i and $f_{i-\varepsilon}$.

Modelling CMO Motion Using a System of Springs

In KR-MARO, each CMO is connected to a system of springs that are arranged in all directions (as shown in Figure 4.9) to handle different combinations of translational and rotational motion. In each direction, there are two virtual springs, i.e. compres-

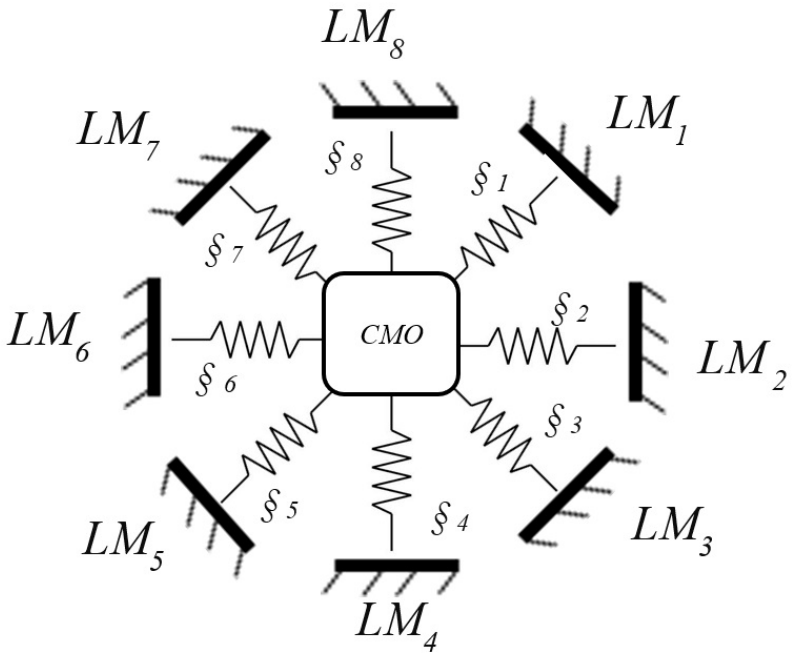


Figure 4.9: KR-MARO is System of Springs

sion and torsion springs, attached to a CMO, and the base of each pair of springs is connected to the same LM. For simplicity, $\oint_w$ is used to denote the two springs together, where $w = 1...8$. The base of $\oint_w$ is connected to 8 landmarks $\{LM_w\}$ that are defined as 8 windows adjacent to this $O_{Cndt_{i,j}}$ in all directions and belong to the background.

To accurately determine if CMO is truly moving and to avoid the restriction of the minimum motion (in pixels per frame) that an object must move in order to be detected, the forces of the system of springs are calculated over a set of frames defined as $set\_F = \{f_i, f_{i\pm(m/3)}, f_{i\pm(m/2)}, f_{i\pm m}\}$ (m is an integer that was experimentally set to be $6 \leq m \leq 15$).

Calculating the kinematic regularization

Initially, all entries in matrices B, O, and D are zeros, while the entries in matrix Γ are ones. Since IALM convergence is q-linear, then the entries in B, O, and D are

Figure 4.10: System of Spring Setup on real frame

near to zero in the first few iterations, and the changes in $F \bullet T - B - D$ are small during these iterations. Hence, $CMOs$ cannot be detected in these early iterations.

However, during the later iterations, the values of B, O, and D increase; hence, $CMOs$ can be located as follows: first, calculate $\check{O} = F \bullet T - B - D$, and $\check{O} \in \mathbb{R}^{r \times n}$ where each of its columns is a vector that contains $CMOs$ found in frame f_i. Second, each column vector in $\check{O}[i]$ is reshaped into a $w \times h$ matrix, and then binarized into a binary mask $\check{f}_i$. Finally, a connected component analysis is applied on each $\check{f}_i$ to locate the groups of non-zero elements. The location of each of these groups in $\check{f}_i$ corresponds to the location of a CMO in f_i.

In KR-MARO, the LMs are selected automatically around each CMO, which is in contrast to Spring-MARO which uses manually predefined LMs as shown in previous section. To show how KR-MARO selects LMs, the following example is instructive. Given a CMO that has a bounding box $[x_{CMO}, y_{CMO}, w_{CMO}, h_{CMO}]$, 8 LMs are defined as windows located next to this CMO in all directions. The size of this window is $w_{LM} \times w_{LM}$; and its top left corner is located on:

1. $LM_1 \rightarrow (x_{CMO} + w_{CMO} + d, \quad y_{CMO} - w_{LM} + d)$

2. $LM_2 \rightarrow (x_{CMO} + w_{CMO} + d, \quad y_{CMO})$

3. $LM_3 \rightarrow (x_{CMO} + w_{CMO} + d, \quad y_{CMO} + h_{CMO} + d)$

4. $LM_4 \rightarrow (x_{CMO}, \quad y_{CMO} + h_{CMO} + d)$

5. $LM_5 \rightarrow (x_{CMO} - w_{LM} - d, \quad y_{CMO} - h_{CMO} - d)$

6. $LM_6 \rightarrow (x_{CMO} - w_{LM} - d, \quad y_{CMO})$

7. $LM_7 \rightarrow (x_{CMO} - w_{LM} - d, \quad y_{CMO} - w_{LM} - d)$

8. $LM_8 \rightarrow (x_{CMO}, \quad y_{CMO} - w_{LM} - d)$

where d is an arbitrary integer value which is used to avoid the overlapping between LMs and CMO in the set_F. After obtaining the location of CMO and LM, the total forces of the system of springs are calculated and the results update $\check{f}_i$. Finally the values in $\check{f}_i$ are used to set the values of Γ. The entries of Γ are zeros at true moving objects when the corresponding value in $\check{f}_i > \alpha$, while Γ entries are arbitrary large values at false detections ($\check{f}_i < \alpha$). Other Γ entries are by default ones. The calculation of Γ is done every IALM iteration for newly detected $CMOs$. To determine whether there are new $CMOs$, the corresponding entries of Γ should be ones for the new CMOs. The Complete KR-MARO algorithm is illustrated in Algorithm 7. The calculation of Γ is described in Algorithm 8

4.6 Results

To evaluate the performance of Spring-MARO and KR-MARO, the experiment in section 3.7 is replicated. The results of this experiment are listed in Tables 4.1and 4.2. By analyzing these tables, the following observations are found: in DARPA

Algorithm 7 Kinematic Regularization and MAtrix Rank Optimization (KRMARO)

Input: F, μ_0, ρ_0, T_0

Output: O

procedure :

  while True **do**

    $F \bullet T_{k+1} = UpdateAlignment(F, T_k)$

    $J_{k+1} = CalculateJacobian(F \bullet T_{k+1})$

    $(U, \Sigma, V) = svd(F \bullet T_{k+1} + \mu^{-1} Y_k - O_k - D_k)$

    $B_{k+1} = U \mathcal{S}_{\mu^{-1}}[\Sigma] V^T$

    $\Gamma_{k+1} = \text{CalculateRegularizationTerm}(F \bullet T_{k+1},\ B_{k+1},\ D_k)$

    $O_{k+1} = \mathcal{S}_{\lambda \mu^{-1} \Gamma_{k+1}}[F \bullet T_{k+1} + Y_k - B_{k+1} - O_k]$

    $D_{k+1} = F \bullet T_{k+1} + \mu^{-1} Y_k - B_{k+1} - O_{k+1}$

    $\triangle t_{k+1} = J_{k+1}^{\dagger}(B_{k+1} + O_{k+1} - F \bullet T_{k+1} - (\mu(1-\eta))^{-1} Y_k)$

    $Y_{k+1} = Y_k + \mu_k (F \bullet T_{k+1} - B_{k+1} - O_{k+1})$

    $\mu_{k+1} = \rho \mu_k$

    $T_{k+1} = T_k + \triangle t_{k+1}$

    if $\frac{\|F \bullet T_{k+1} - B_{k+1} - O_{k+1} - D_{k+1}\|_f^2}{\|F \bullet T_{k+1}\|_f^2} < \epsilon$ **then**

      if $d(\triangle t_{k+1}, \triangle t_k) < \varepsilon$ and $|\triangle t_{k+1}| < \zeta$ **then**

        Break

      else

        $B_{k+1} = B_{k-\kappa}$

        $O_{k+1} = O_{k-\kappa}$

        $D_{k+1} = D_{k-\kappa}$

        $Y_{k+1} = Y_{k-\kappa}$

        $\mu_{k+1} = \mu_{k-\kappa}$

      end if

    end if

  end while

end procedure

Algorithm 8 Calculate the kinematic regularization Γ

Input: $F \bullet T$, B, D

Output: Γ

procedure CalculateRegularizationTerm

    $\check{O} = Binarization(F \bullet T - B - D)$

    for `(i=1 ; i <= FramesCount ; i++)` **do**

        $\check{f}_i = vec^{\dagger}(\check{O}[i])$

        Obtain $CMOi, j$ in f_i using connect component analysis

        for `(j=1 ; j<=ComponentsCount ; j++)` **do**

            $\upsilon = \Sigma_{w=1}^{8} CSF(CMO_{i,j}, LM_w)$

            $\varphi = \Sigma_{w=1}^{8} TSF(CMO_{i,j}, LM_w)$

            Update $(\check{f}_i)$ with $\upsilon + \varphi$ at location of $CMO_{i,j}$

        end for

        $TF[i] \leftarrow vec(f_i)$

    end for

    if $TF > \alpha$ **then** $\Gamma = 0$

    else$\Gamma = \Omega$ *// Ω is an arbitrary large value*

    end if

end procedure

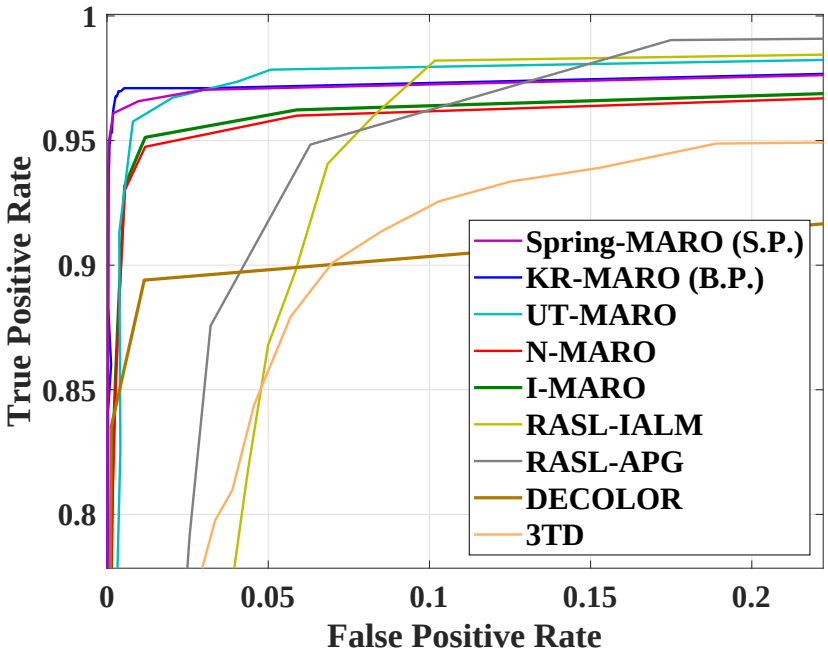


Figure 4.11: ROC curves for Spring-MARO, KR-MARO, UT-MARO, N-MARO, I-MARO, RASL-IALM, RASL-APG, DECOLOR, and 3TD on DARPA VIVID

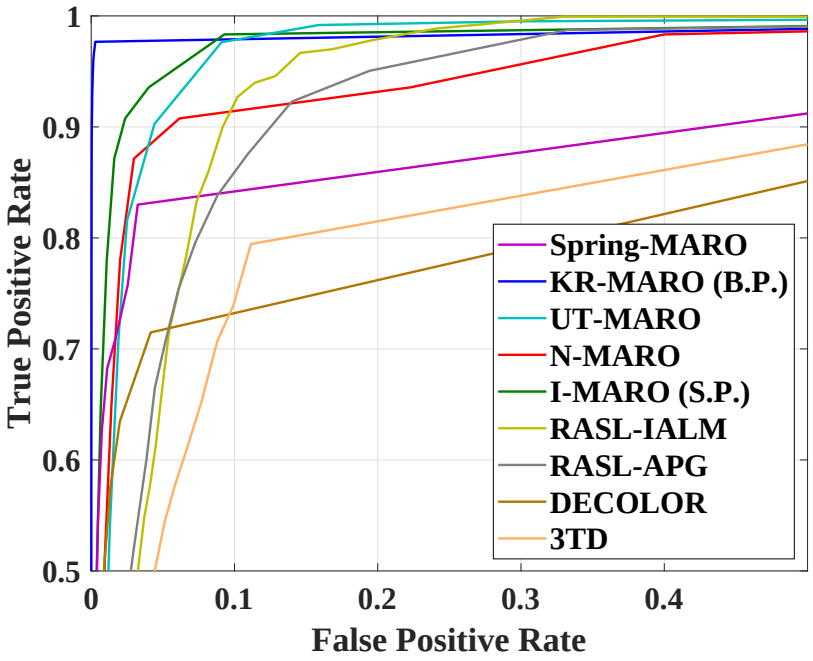


Figure 4.12: ROC curves for Spring-MARO, KR-MARO, UT-MARO, N-MARO, I-MARO, RASL-IALM, RASL-APG, DECOLOR, and 3TD on UCF aerial action

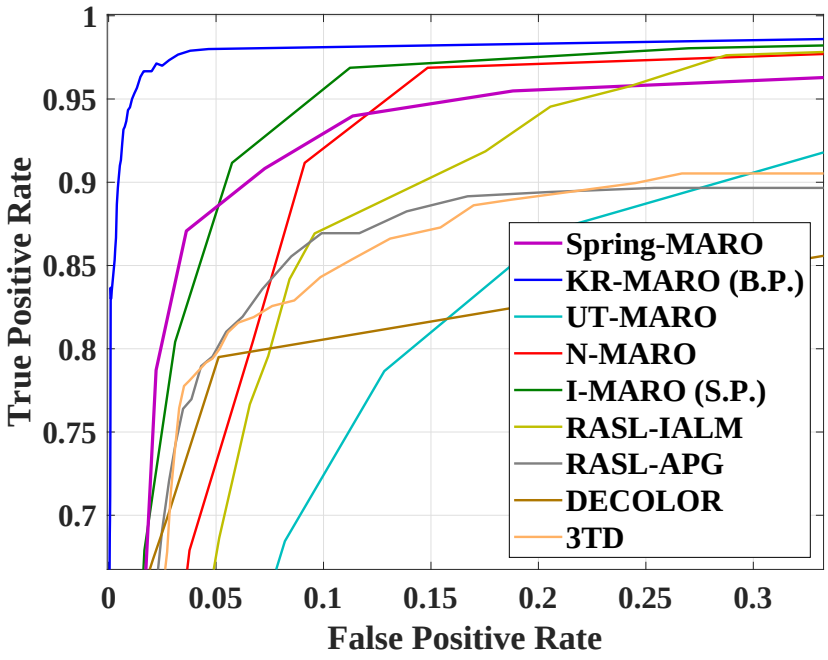


Figure 4.13: ROC curves for Spring-MARO, KR-MARO, UT-MARO, N-MARO, I-MARO, RASL-IALM, RASL-APG, DECOLOR, and 3TD on VIRAT

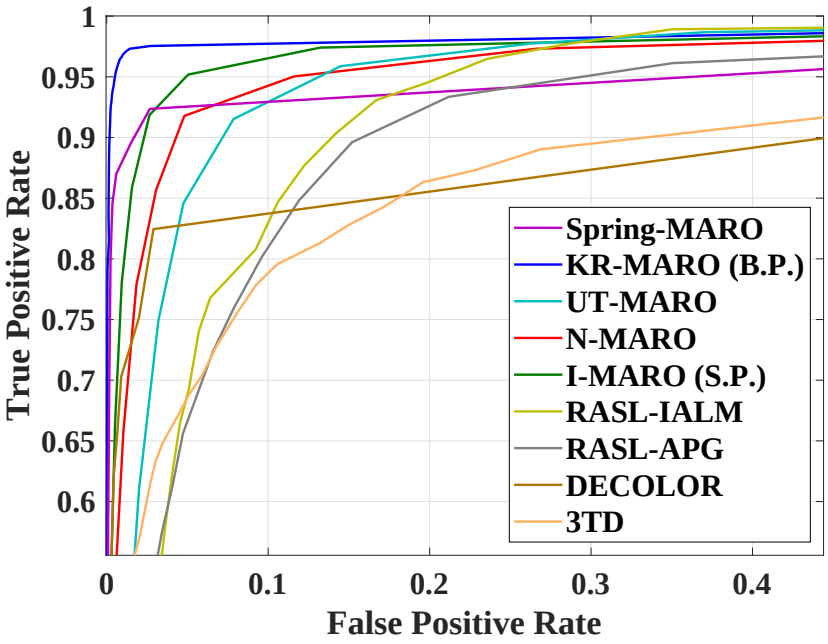


Figure 4.14: Average ROC curves for Spring-MARO, KR-MARO, UT-MARO, N-MARO, I-MARO, RASL-IALM, RASL-APG, DECOLOR, and 3TD

VIVID dataset, KR-MARO has FPR (0.3%) while achieving high TPR (97%). Although MARO, N-MARO, UT-MARO and Spring-MARO have high TPR, their false detections are noticeable. RASL-IALM, and RASL-APG and 3TD are able to achieve good TPR (94%, 92.5% and 92.3% in order), but they suffer from high false detections; FPR is 6%, 6.5% and 11.3%, respectively. DECOLOR has very low TPR, 88.3%, with FPR equal to 1.6%.

In the UCF dataset, there is a shaky motion of the camera platform and the size of the moving objects is very small. KR-MARO achieves the highest true detection: TPR is 97.5% and false detections rate (FPR) is 0.2%. On the other hand, FPR of MARO, N-MARO, RASL-IALM, RASL-APG are high compared to the proposed method: 6.4%, 12.3%, 13.4%,14.4%, respectively. UT-MARO and Spring-MARO have low false detections; however, this is at the expense of true detections. DECOLOR and 3TD have high FPR and low TPR.

Although the shakiness in the VIRAT dataset is high, KR-MARO maintains the best true detections (TPR is 95%) with very low false detections (FPR is 1%). MARO, N-MARO and RASL-IALM get high true detections (TPR is 94.4%, 93% and 93%, respectively), yet false detections are high (FPR is 7.4%, 13.6% and 13.7%, respectively). The accuracy of UTMARO, Spring-MARO, RASL-APG, DECOLOR and 3TD severely suffers in this dataset, where their TPRs are 86.6%, 90%, 86.7%, 78.4% and 87%, respectively, and their FPRs are 25%, 5.3%, 25%, 8.7% and 28.4%, respectively. The complete relationship between TPR and FPR of the compared methods is depicted via the ROC curves in figures 4.11 to 4.14. For sample results, please refer to figures 4.15 to 4.23. For the full video sequence, please visit our website: https://phdthesisvisualresults.weebly.com/

Table 4.1: Quantitative evaluation (TPR and FPR) using DARPA VIVID, UCF aerial action and VIRAT datasets

	Dataset	Spring-MARO		KR-MARO		UT-MATO		N-MARO		I-MARO	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest 1	96.5%	**0.09%**	**98%**	1%	97%	1%	96%	0.3%	96%	0.3%
	Egtest 2	**100%**	0.3%	99%	**0.2%**	97%	0.1%	99%	0.7%	99%	0.7%
	Egtest 5	**93%**	0.9%	92%	**0.4%**	91%	0.6%	90%	1%	90%	1%
	Egtest 3	**98%**	0.7%	**98%**	**0.2%**	**98%**	0.9%	97%	0.9%	97%	0.9%
	Average	**96.7%**	0.49%	97%	**0.3%**	95.7%	0.6%	95.5%	0.7%	95.5%	0.7%
UCF	Action 1	87%	3%	**98%**	**0.4%**	93%	4%	86%	10%	95%	10%
	Action 2	94%	4%	**97%**	**0.2%**	81%	4%	96%	20%	98%	4%
	Action 3	60%	1%	**96%**	**0.3%**	95%	5%	90%	7%	96%	5%
	Average	80%	2.6%	**97%**	**0.2%**	89.6%	4.3%	90.6%	12.3%	96.4%	6.4%
VIRAT	Flight2tape1_2	91%	2%	**98%**	**0.1%**	80%	40%	94%	9%	95%	4%
	Flight2tape2_1	85%	10%	**90%**	**2%**	90%	25%	91%	23%	92%	10%
	Flight2tape3_2	94%	4%	**98%**	**0.1%**	90%	10%	94%	9%	96%	8%
	Average	90%	5.3%	**95%**	**0.7%**	86.6%	25%	93%	13.6%	94.4%	7.4%
	Total Average	88.9%	2.8%	**97%**	**0.4%**	90.6%	9.9%	93%	8.8%	95.4%	4.8%

Table 4.2: Quantitative evaluation (TPR and FPR) using DARPA VIVID, UCF aerial action and VIRAT datasets

	Dataset	KR-MARO		RASL-IALM		RASL-APG		DECOLOR		3TD	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest 1	98%	1%	95%	6%	95%	7%	92%	1%	95%	5%
	Egtest 2	99%	0.2%	98%	6%	94%	13%	92%	1.5%	93%	12%
	Egtest 5	92%	0.4%	88%	7%	87%	4%	85%	1.7%	87%	20%
	Egtest 3	98%	0.2%	95%	5%	94%	2%	84%	2%	94%	8%
	Average	**97%**	**0.3%**	**94%**	**6%**	**92.5%**	**6.5%**	**88.3%**	**1.6%**	**92.3%**	**11.3%**
UCF	Action 1	98%	0.4%	94%	17%	92%	20%	72%	11%	85%	20%
	Action 2	97%	0.2%	95%	13%	92%	11%	71%	7%	85%	15%
	Action 3	96%	0.3%	92%	10%	91%	12%	68%	5%	70%	10%
	Average	**97%**	**0.2%**	**93.7%**	**13.4%**	**91.7%**	**14.4%**	**70.4%**	**7.7%**	**80%**	**15%**
VIRAT	Flight2tape1_2	98%	0.1%	94%	9%	80%	40%	70%	5%	78%	35%
	Flight2tape2_1	90%	2%	91%	23%	90%	25%	80%	12%	93%	30%
	Flight2tape3_2	98%	0.1%	94%	9%	90%	10%	85%	9%	90%	20%
	Average	**95%**	**0.7%**	**93%**	**13.7%**	**86.7%**	**25%**	**78.4%**	**8.7%**	**87%**	**28.4%**
	Total Average	**97%**	**0.4%**	**93.5%**	**11%**	**90.3%**	**15.3%**	**79%**	**6%**	**86.4%**	**18.2%**

4.7 Conclusion

This chapter proposes two PCP based methods, namely Spring-MARO and KR-MARO, that integrate a regularization term to enhance the detection accuracy. Spring-MARO uses a simple spring model, based on compression spring, to build the regularization term and the bases of the spring is a manually selected landmark. KR-MARO proposes a kinematic regularization term based on an advanced spring model that combines compression and torsion springs to handle translational and rotational motion. Moreover, KR-MARO solves the PCP formulation via IALM with backtracking behaviour and the inexact Newton method, which is in contrast to Spring-MARO that uses IALM and the weighted Newton method.

The performance evaluation of Spring-MARO shows a great success with DARPA dataset, but its performance is badly affected in the rest of datasets. However, KR-MARO keeps a stable performance for all datasets, e.g. high true detection rates and very low false detection rates. Moreover, KR-MARO outperforms the current state-of-the-art methods: I-MARO, N-MARO, UT-MARO, RASL-IALM, RASL-APG, DE-COLOR, and 3TD. This superior performance of KR-MARO is credited mainly to its kinematic regularization term.

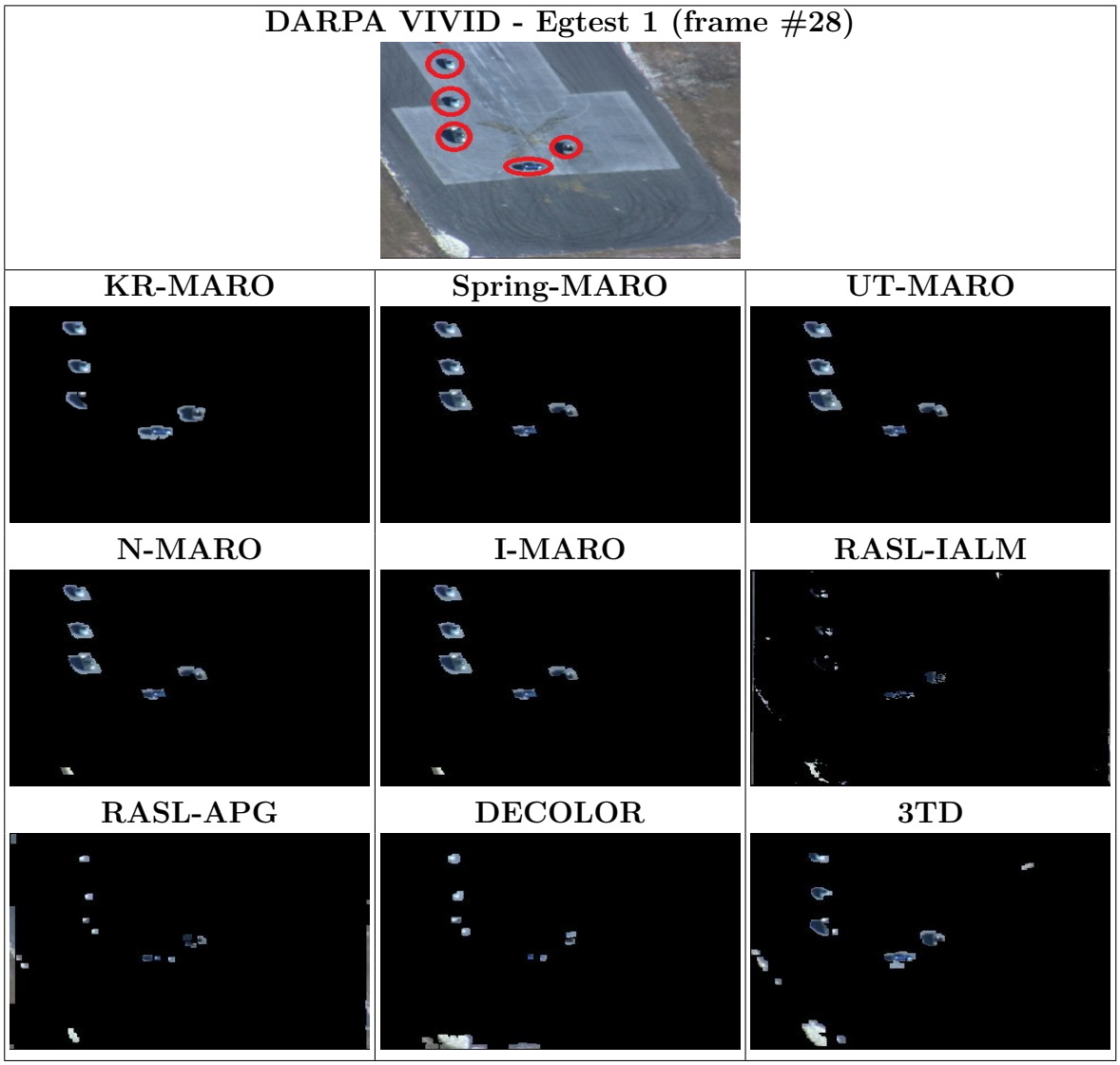


Figure 4.15: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on DARPA VIVID

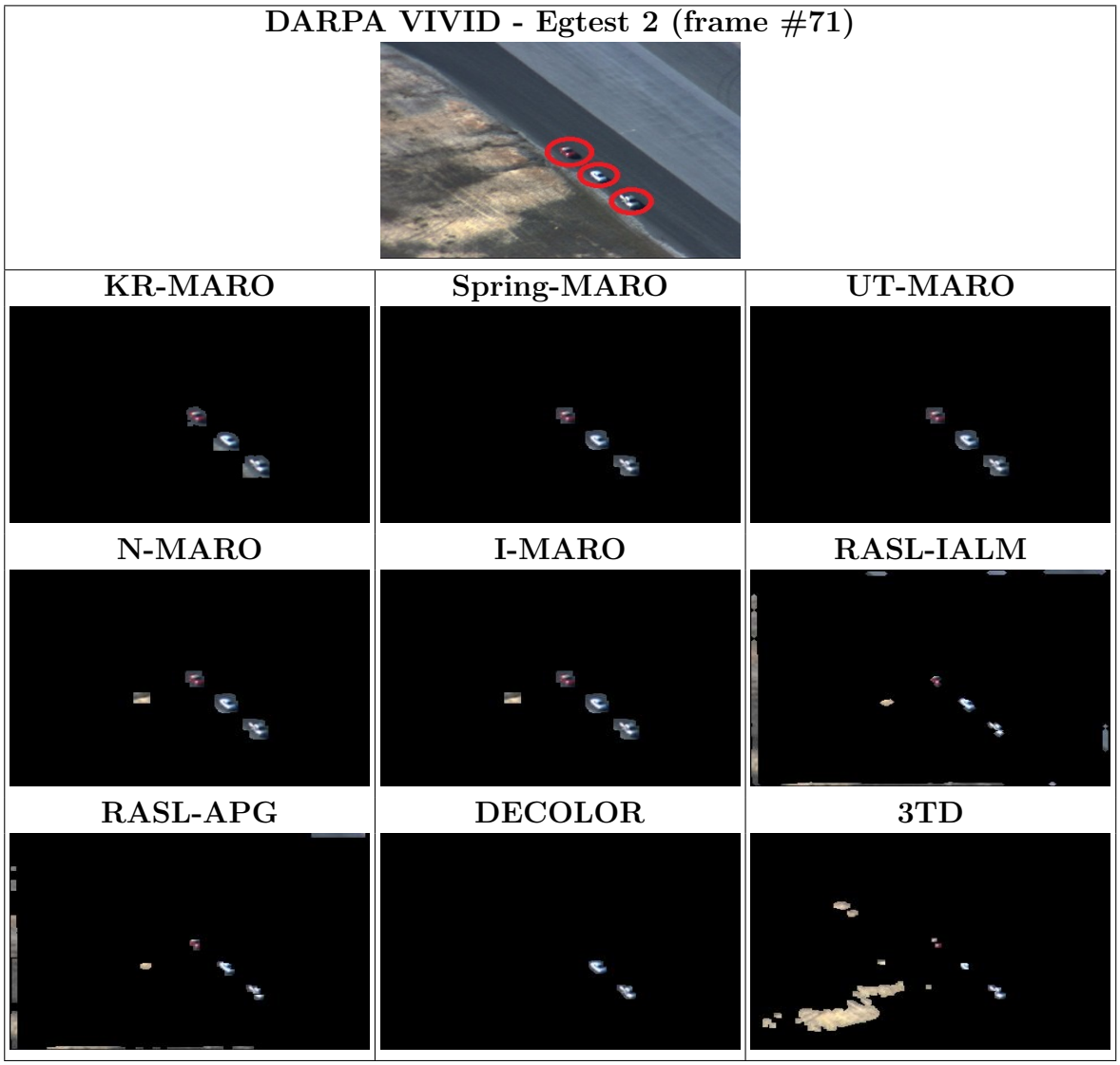


Figure 4.16: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on DARPA VIVID continued

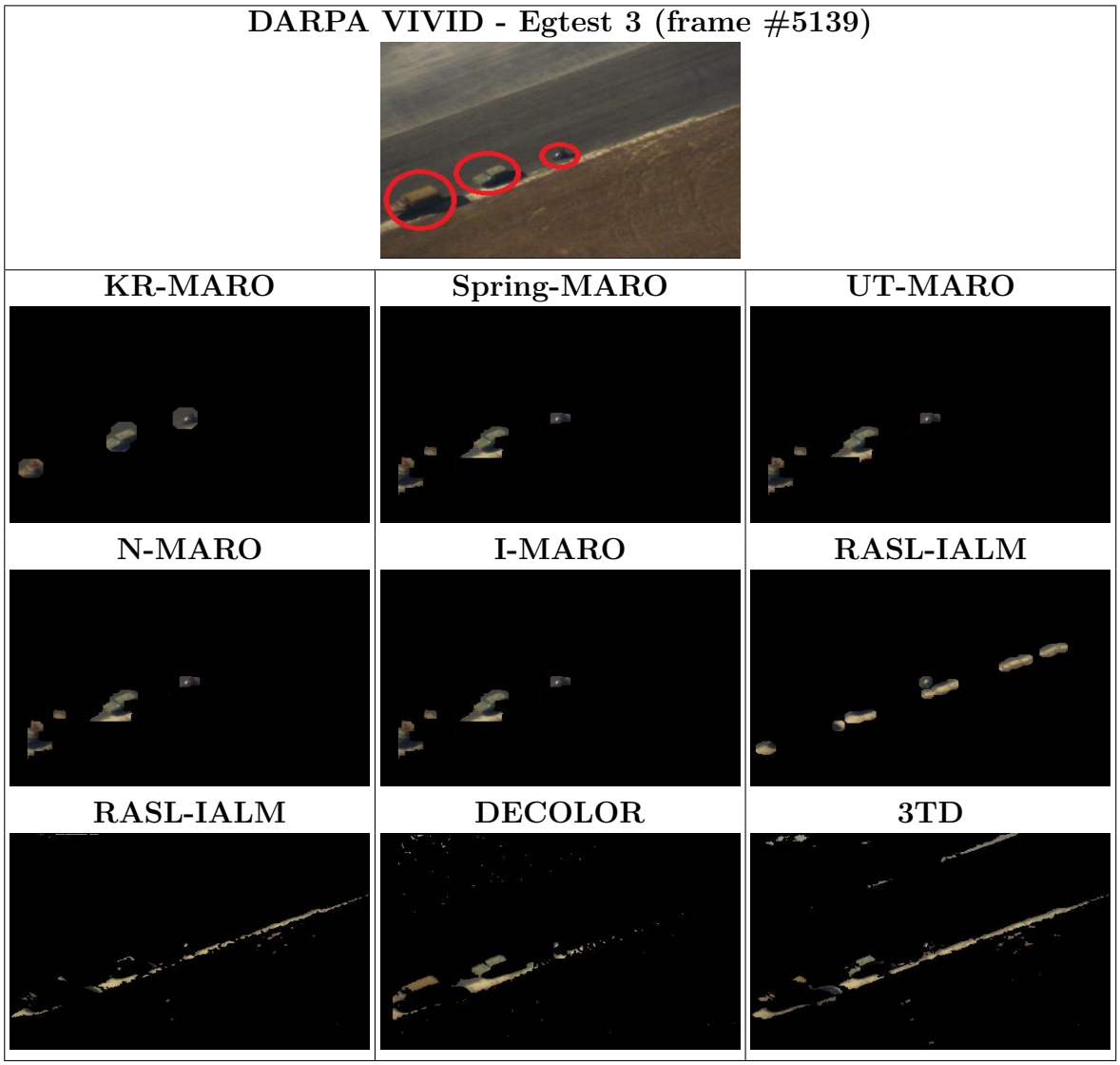


Figure 4.17: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on DARPA VIVID continued

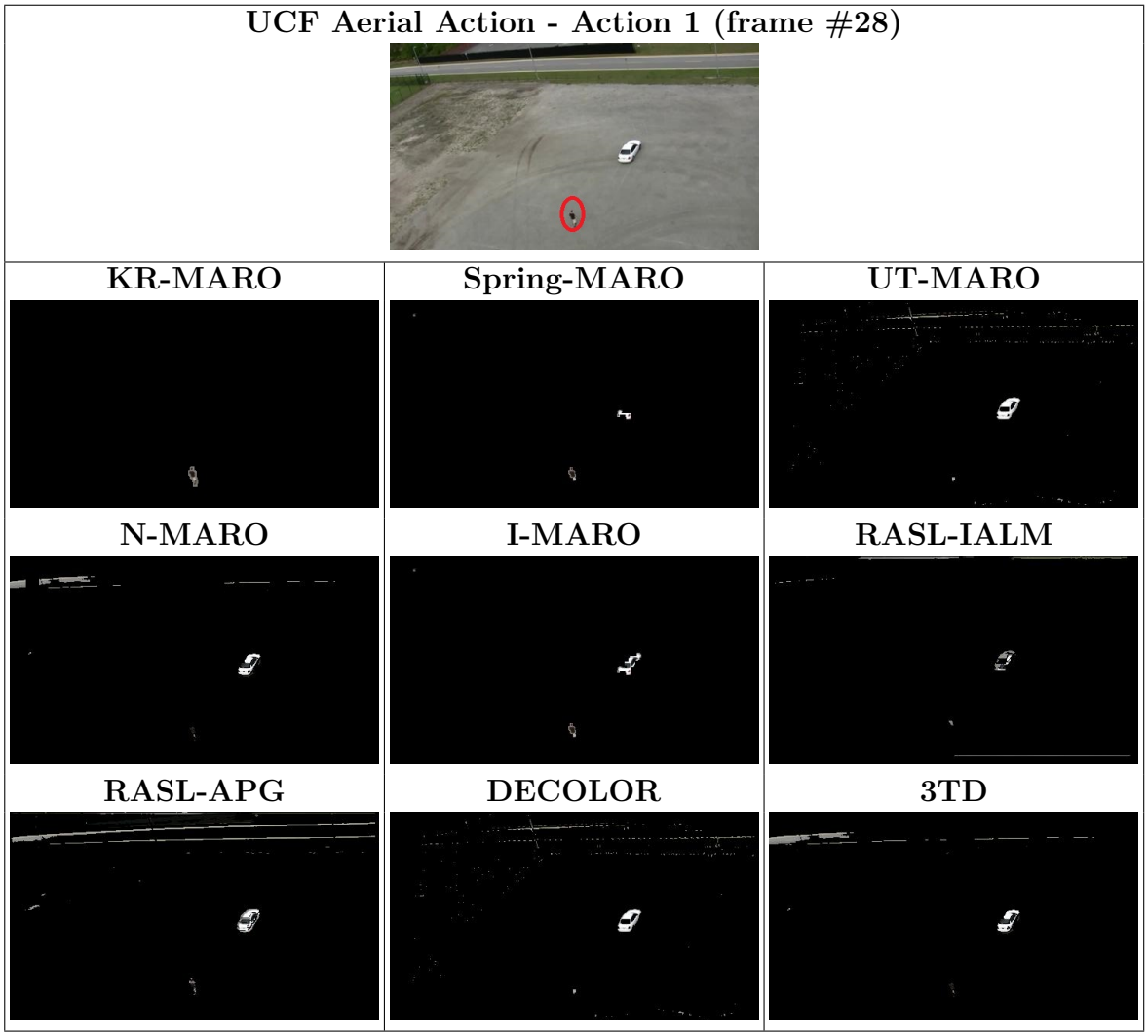


Figure 4.18: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on UCF aerial action

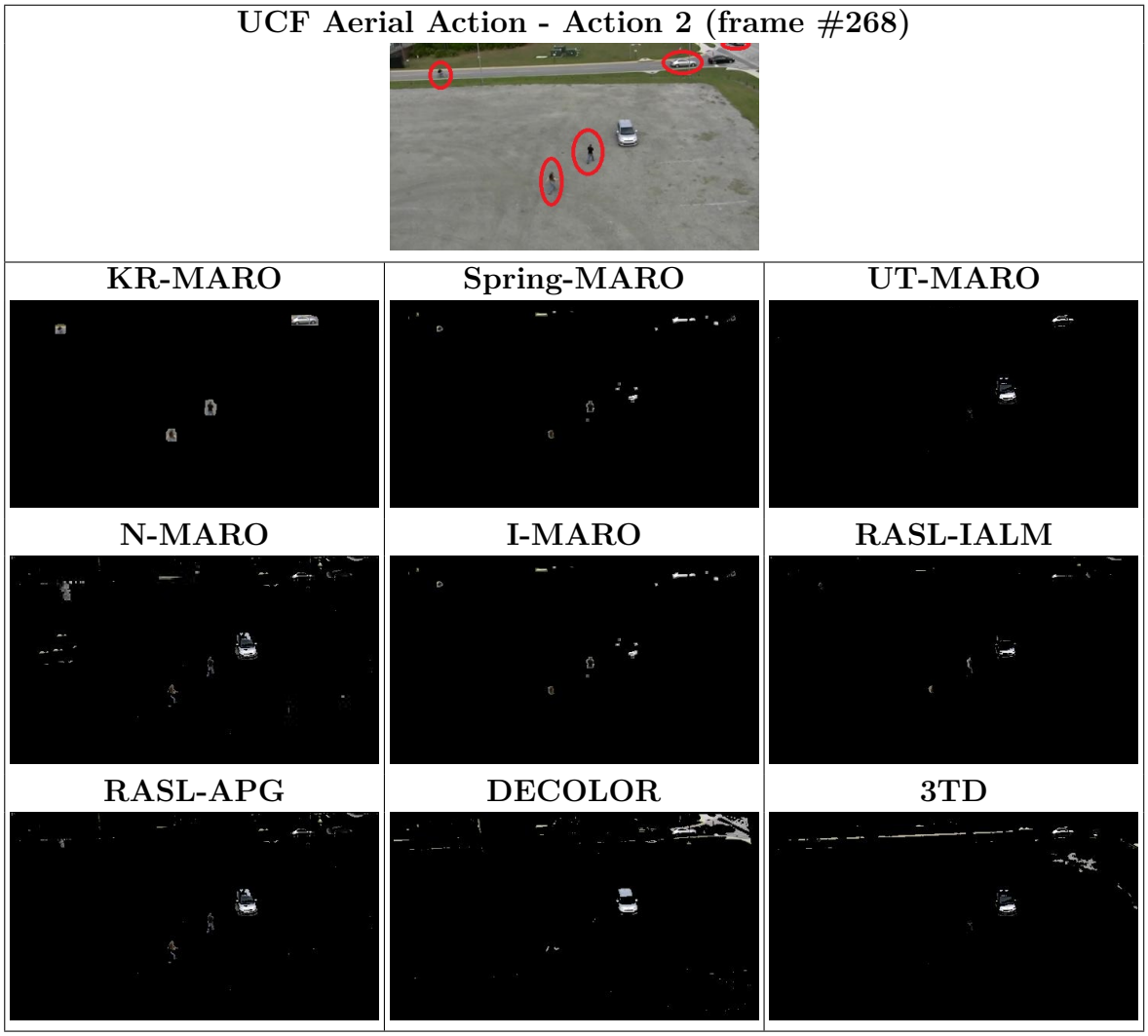


Figure 4.19: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on UCF aerial action continued

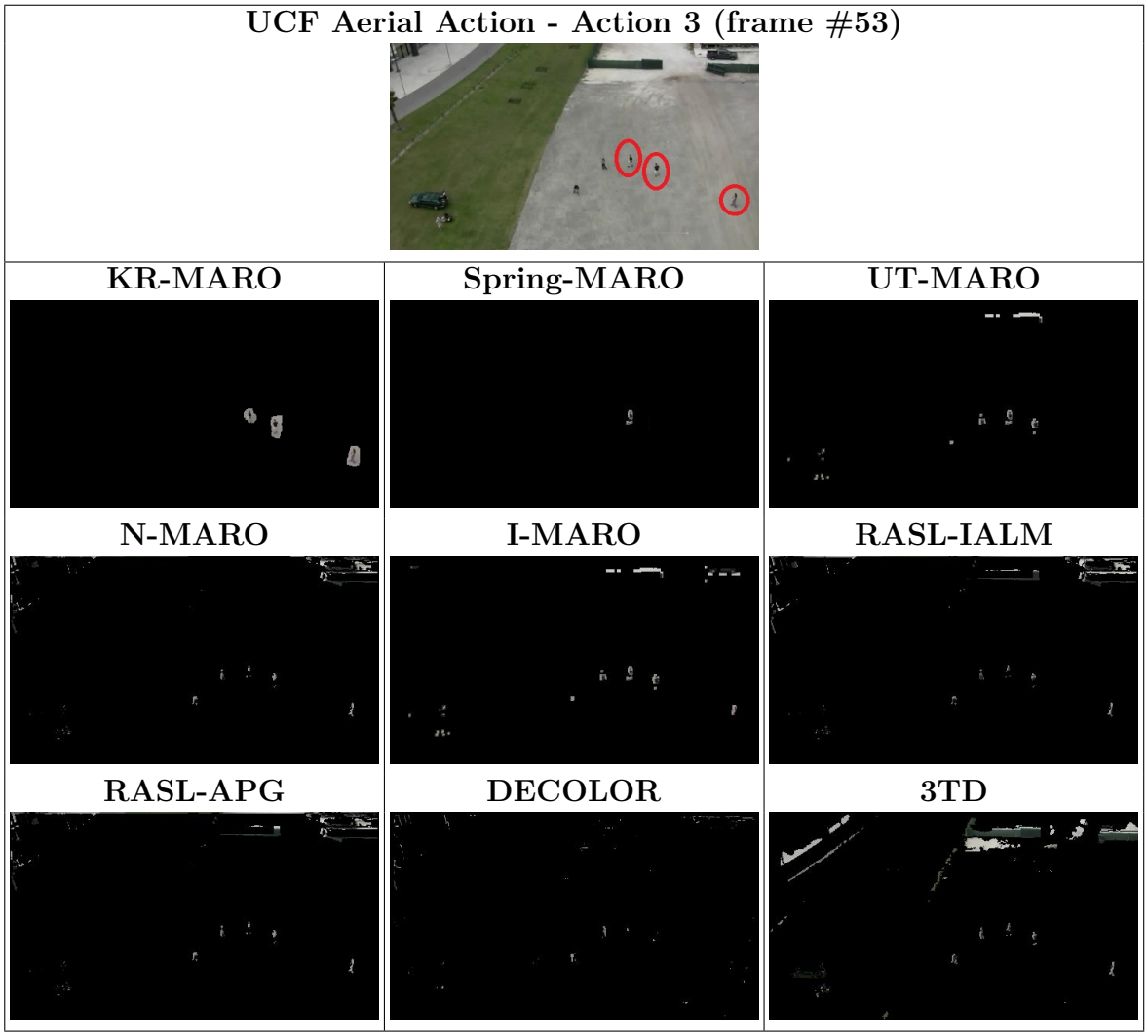


Figure 4.20: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on UCF aerial action continued

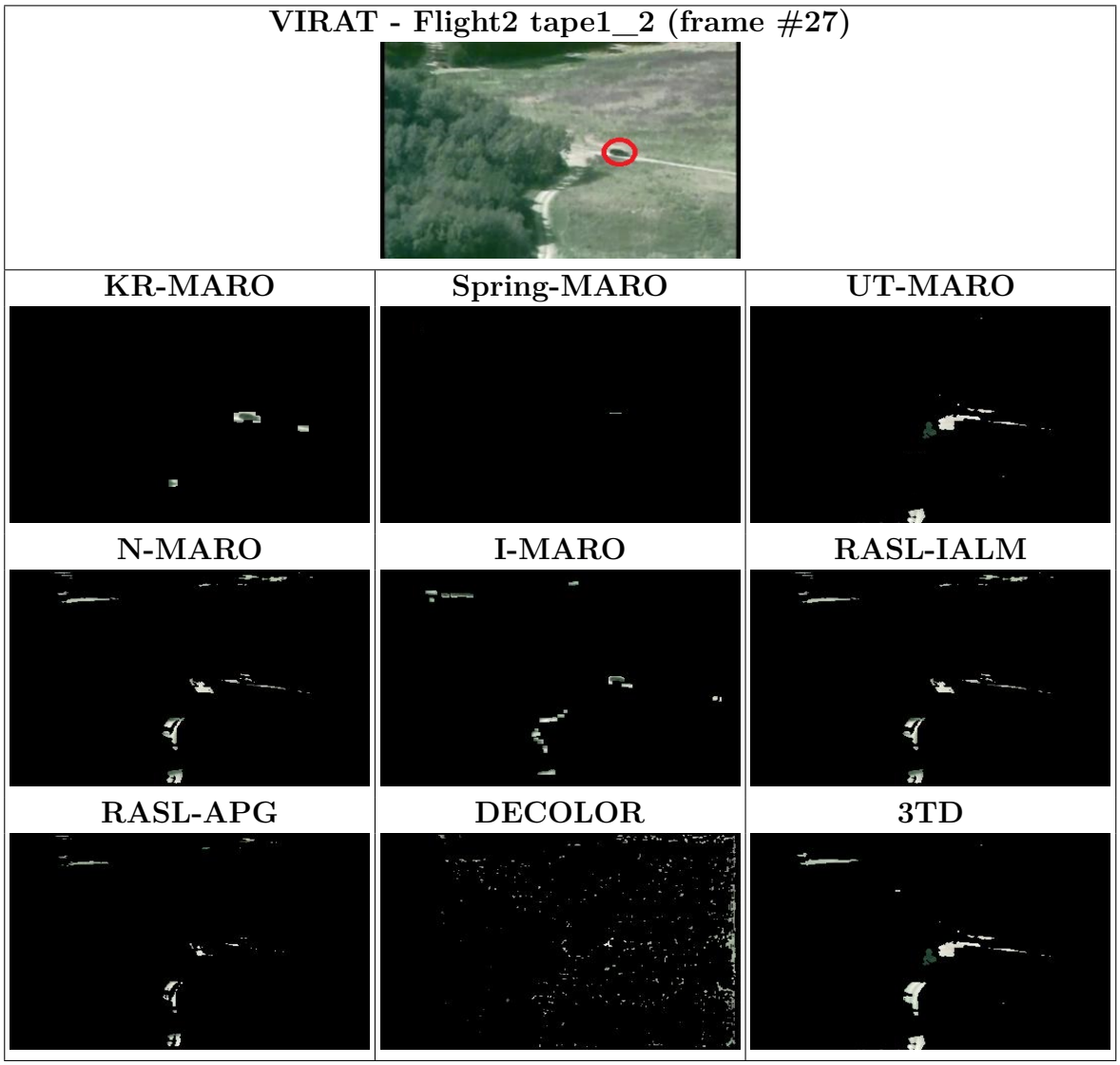


Figure 4.21: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on VIRAT

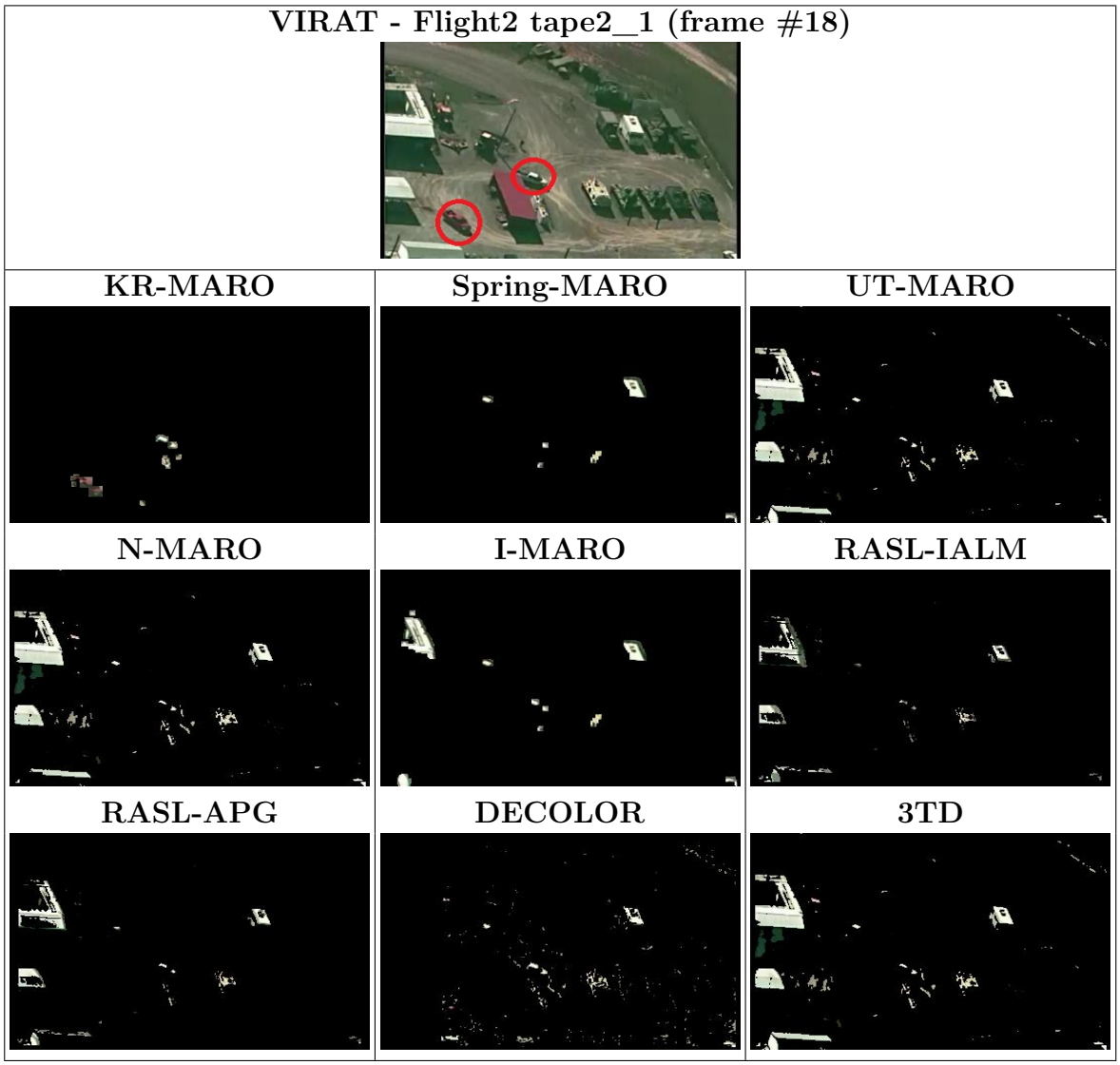


Figure 4.22: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on VIRAT continued

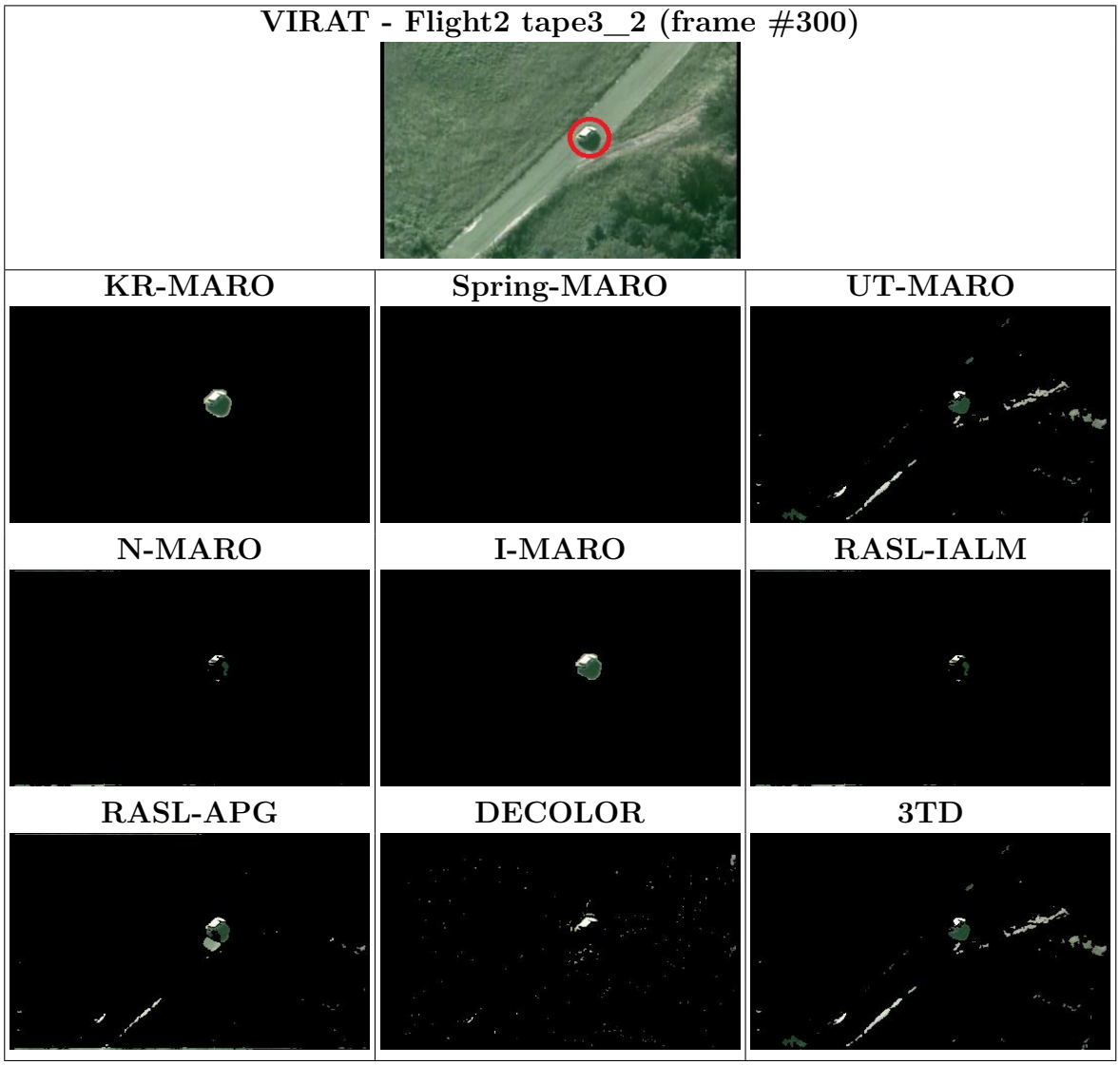


Figure 4.23: Sample results of Spring-MARO , KR-MARO, and PCP-based detection methods on VIRAT continued

Chapter 5

Sequential-based Principle Component Pursuit

5.1 Introduction

Previous chapters study how to achieve accurate true detections and reduce false detections. The result of these studies is KR-MARO method (presented in chapter 4) that radically reduces false detection rates radically and keep true detection rates very high. However, the presented methods so far, including KR-MARO, cannot provide adequate real-time performance, the key requirement in aerial imagery applications, as a typical consequence of their batch-based processing. This chapter investigates how to extend PCP concepts to fulfill the real-time performance requirement.

5.2 Chapter Contributions

The contributions of this chapter are: (1) modelling the background in a video as a subspace which lies in a low-dimension subspace (contrary to presented methods in

previous chapter that models the background as a low-rank matrix), (2) proposing a new PCP formulation based on local Null-Spaces, (3) integrating the kinematic regularization term into the new PCP formulation. These contributions are presented through two methods: Local Null-Space Pursuit (LNSP) and Kinematic Regularization with Local Null Space Pursuit (KR-LNSP). These methods have been submitted to two journal papers as follows:

- Agwad ElTantawy and Mohamed S. Shehata. "LNSP: Local Null-Space Pursuit for Real-Time Detection in Aerial Surveillance" *Under revision in IEEE transactions on image processing*
- Agwad ElTantawy and Mohamed S. Shehata. "A Sequential-based PCP method for Ground-Moving Objects Detection from Aerial Videos" *Under revision in Journal of Signal, Image, and Video Processing*

The results in this chapter are replicated from the former papers as well.

5.3 Null Space Pursuit

Given a noisy data observed in a finite time horizon such that data forms a matrix with rank R, and according to PCP, this matrix can be decomposed into a core structure and a sparse matrix. This core structure is a low-rank matrix where its rank is r and $r \ll R$. Similarly, the background in a finite sequence of frames can be modelled as a low-rank matrix; hence, the moving objects can be detected as sparse corrupting these frames. Unfortunately, the former modelling of the background is the reason of the batch processing behaviour in PCP detection methods. This is because a low-rank matrix can be obtained only from multiple matrices; hence, input frames have to be buffered to form a matrix.

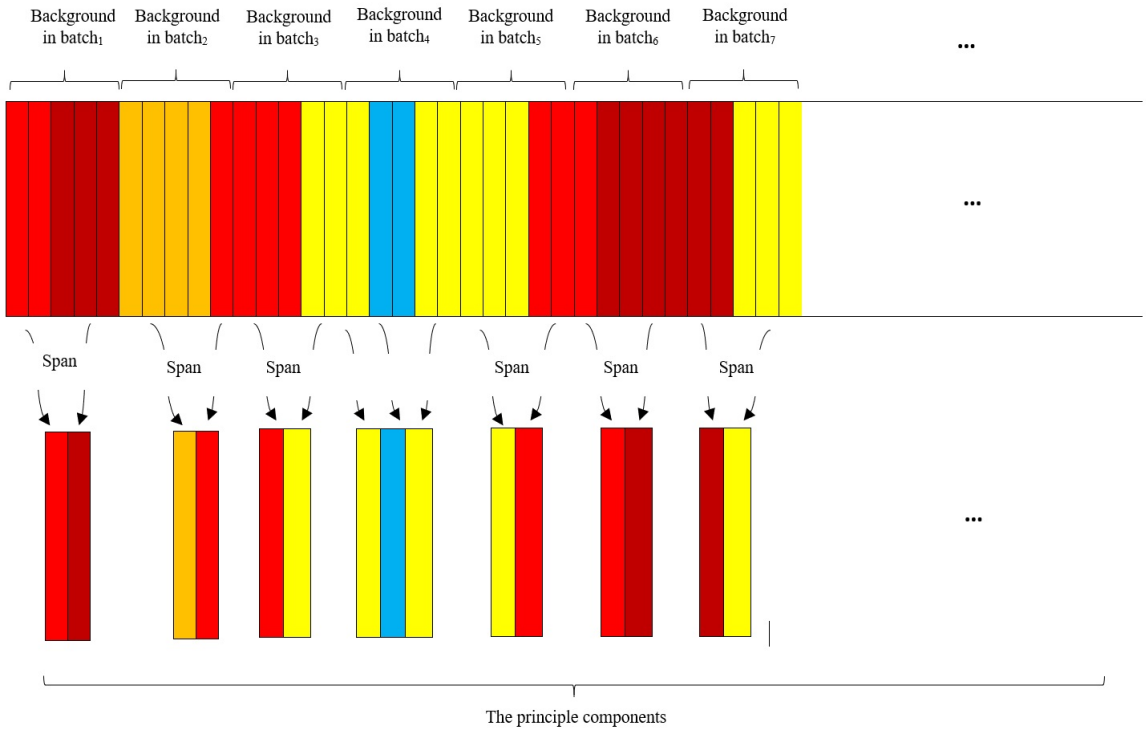


Figure 5.1: Principle components of the background

To overcome this problem, the former modelling of the background is extended as explained in the following. Modelling the background B as a low-rank matrix means the background vectors $\{vec(b_1), vec(b_2), ..., vec(b_i)\}$ in frames $\{vec(f_1), vec(f_2), ..., vec(f_i)\}$ are linearly dependent. Hence, the background $B \in \mathbb{R}^{m \times n}$ spans a few principal components $\in \mathbb{R}^{m \times \varsigma}$ where $\varsigma \ll n$. Based on this fact, the background in the whole video stream can be modelled as a subspace $\mathcal{B}$ (see definition 5.3.1) which spans (or lies in) a low-dimension subspace (called principal components space $\mathcal{PC}$), i.e. $span(\mathcal{PC}) = \mathcal{B}$, as shown in Figure 5.3. The background vectors $\{vec(b_1), vec(b_2), vec(b_3), ...\} \in \mathcal{B}$ can be grouped into $\{G_1, G_2, G_3, ...\}$. $G_\rho \subset \mathcal{B}$ is a group of colinear vectors i.e. $\{\forall\ vec(b_x), vec(b_y) \in G_\rho |\ span(vec(b_x)) = vec(b_y); x, y = 1, 2, 3, ..., sizeof(G_\rho)\}$. In the case of video processing, since the variation across consecutive frames is small, then the degree of freedom between these frames is also small. Hence, the background in a sequence of consecutive frames ($i - \varepsilon$ to i) is most likely to exist in one G_ρ, there-

fore, G_ρ can be defined as $\{vec(b_{i-\varepsilon}), ..., vec(b_i)\}$ where i is the frame index, ε is a small integer positive value. Consequently, from the definition of null space [77], the orthonormal complement of the null space $\mathcal{N}_{i-\varepsilon}$ of $vec(b_{i-\varepsilon})$ nullifies $vec(b_i)$:

$$\mathcal{N}_{i-\varepsilon}^{\top} vec(b_i) = 0 \tag{5.1}$$

where the backgrounds $vec(b_{i-\varepsilon})$ and $vec(b_i)$ in frames $vec(f_i)$ and $vec(f_{i-\varepsilon})$ are colinear; $\mathcal{N}_{i-\varepsilon} = \{x \in \mathbb{R}^m |\ x^{\top} vec(b_{i-\varepsilon}) = 0\}$. From the above, the moving objects can be detected by nullifying $vec(b_i)$ using $\mathcal{N}_{i-\varepsilon}$.

Definition 5.3.1. *Assume $\mathcal{K}$ is a subset of a vector space $\mathcal{V}$, $\mathcal{K}$ is called a subspace of $\mathcal{V}$ when the zero vector $0 \in \mathcal{K}$, $\mathcal{S}$ is closed under addition, and $\mathcal{K}$ is closed under multiplication by scales, as described below:*

1. *$\vec{0} \in \mathcal{K}$*
2. *if $\vec{u}$ and $\vec{v} \in \mathcal{K}$, then $\vec{u} + \vec{v} \in \mathcal{K}$*
3. *if $\vec{u} \in \mathcal{K}$ and α is a scaler, then $c\vec{u} \in \mathcal{K}$*

5.4 LNSP: Local Null Space Pursuit

The LNSP method uses the concept introduced above to provide a novel problem formulation to detect the moving objects sequentially.

5.4.1 Problem Formulation

A typical frame in aerial surveillance is decomposed into a background and moving objects:

$$vec(f_i) = vec(b_i) + vec(o_i) \tag{5.2}$$

Then From Eq. 5.2, Eq. 5.1 can be written as:

$$\mathcal{N}_{i-\varepsilon}^{\top} vec((f_i - o_i) \bullet T_i) = 0 \tag{5.3}$$

Similar to the original PCP formulation, the moving objects are modelled as sparse that can be obtained by minimizing $\ell_1 - norm$ as follows:

$$\begin{aligned} \min_{o_t} \quad & \|vec(o_t)\|_1 \\ s.t. \quad & \|\mathcal{N}_{t-1}^{\top} vec((f_i - o_i) \bullet T_i)\|_2 = 0 \end{aligned} \tag{5.4}$$

where o_t is the moving objects; $\|.\|_1$ and $\|.\|_2$ are $\ell_1 - norm$ and $\ell_2 - norm$, respectively.

However, Eq. 5.4 is based on the global null space, and calculating this null space is very time consuming. To clarify these points, assume the following scenarios:

- Given a vector (V) with size 76800, then V is divided into smaller non-overlapping vectors with size 100, i.e. $\{v_1, v_2, \ldots., v_{768}\}$. We end-up with 768 small vectors. The time required to calculate the null space of each small vector v_i is 0.0001 seconds on average, (according to the computer specifications described in 3.7); then, the time required to calculate the null spaces for all small vectors $\{v_1, v_2, \ldots., v_{768}\}$ is 0.07 seconds on average.

- Divide V into small vectors with size 400, i.e. $\{v_1^{'}, v_2^{'}, \ldots., v_{192}^{'}\}$, and we end-up with 192 non-overlapping small vectors. The time required to calculate the null space of each small vector $v_i^{'}$ is 0.005 seconds on average, then the time required to calculate the null spaces for all small vectors $\{v_1^{'}, v_2^{'}, \ldots., v_{768}^{'}\}$ is 0.96 seconds.

For the cases above, the time required to calculate the null space is directly proportional to the size of the vector.

LNSP calculates multiple local null spaces, as depicted in Figure 5.2. LNSP divides

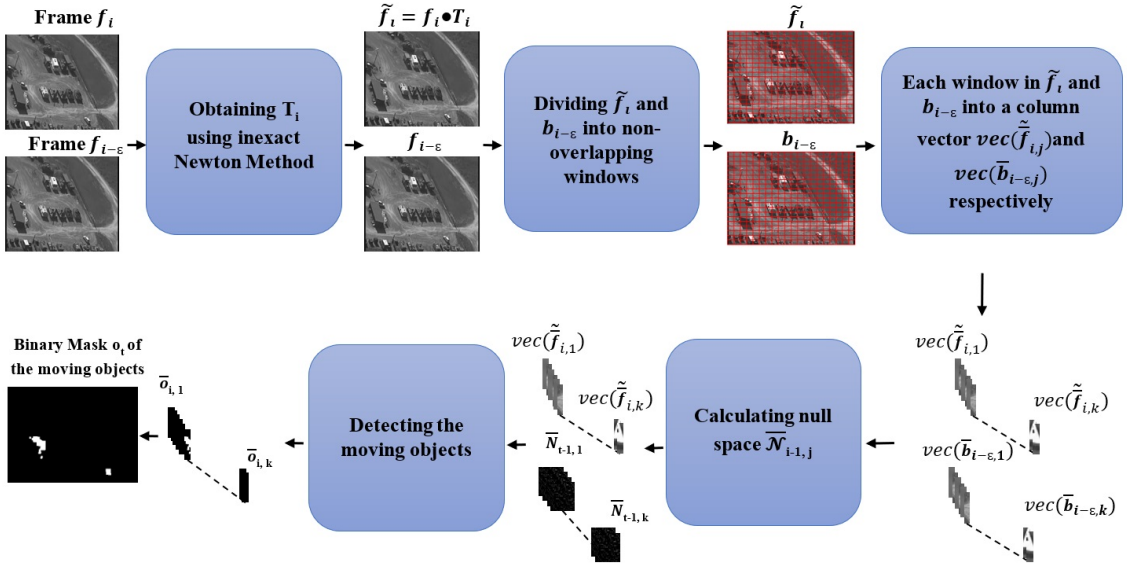


Figure 5.2: Detecting moving objects based on multiple local null spaces

the background into non-overlapping windows, and calculates a null space for each $\bar{b}_{i-\varepsilon,j}$ (where $\bar{b}_{i-\varepsilon,j} \in \mathbb{R}^w$ corresponds to a window and w is the window size, j is the index of the window). Consequently, the detection problem is formulated as follows:

$$\begin{aligned} \underset{vec(o_i)}{min} \quad & \textstyle\sum_{j=1}^{k}(\|vec(\bar{o}_{i,j})\|_1) \\ s.t. \quad & \textstyle\sum_{j=1}^{k}(\|\overline{\mathcal{N}}_{i-\varepsilon,j}^{\top} vec((\bar{f}_{i,j} - \bar{o}_{i,j}) \bullet T_i)\|_2) = 0 \end{aligned} \tag{5.5}$$

where $vec(\bar{o}_{i,j})$ represents the moving objects in $vec(\bar{f}_{i,j})$ of the current frame $vec(\bar{o}_{i,j})$ and $vec(\bar{f}_{i,j}) \in \mathbb{R}^w$, and k is the number of windows.

5.4.2 Problem Formulation Solution

To solve Eq. 5.5, it is rewritten in the Lagrange form as:

$$\mathcal{L}(vec(o_i)) = \sum_{j=1}^{k}(\|vec(\bar{o}_{i,j})\|_1) - \lambda \sum_{j=1}^{k}(\|\overline{\mathcal{N}}_{i-\varepsilon,j}^{\top} vec((\bar{f}_{i,j} - \bar{o}_{i,j}) \bullet T_i)\|_2) \tag{5.6}$$

where λ is the Lagrange multiplier. The closed-form solutions to calculate T_i and o_i are illustrated in the next subsections.

The closed-form solution of T_i

LNSP proposes usage of the inexact Newton method for a rapid and an accurate calculation of T_i. The inexact Newton method generates a sequence of $\{\triangle t_i\}$ to approximate T_i^* (the optimal value of T_i) by searching directions that satisfy the inexact Newton method's condition [102]. Hence, T_i^* must satisfy the following:

$$\|(b_{i-\varepsilon} - (b_i \bullet T_i)) + J_i \triangle t_i\| \leq \eta \|(b_{i-\varepsilon} - (b_i \bullet T_i))\| \tag{5.7}$$

where η is a forcing term; and J is the Jacobian matrix. Since b_i is unknown in the current frame, matrix T_i is calculated between f_i and $f_{i-\varepsilon}$, for simplicity. Accordingly, Eq. 5.7 will be:

$$\|(f_{i-\varepsilon} - (f_i \bullet T_i)) + J_t \triangle t_i\| \leq \eta \|(f_{i-\varepsilon} - (f_i \bullet T_i))\| \tag{5.8}$$

The forcing term η is calculated as in [103]:

$$\eta = \gamma(\frac{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega}\|}{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega-1}\|})^{\alpha} \tag{5.9}$$

the superscript Ω refers to the iteration index in the inexact Newton method, $\gamma \in [0, 1]$ and $\alpha \in (1, 2]$. From Eq. 5.8, $\triangle t^i$ is obtained as follows:

$$\triangle t^{\Omega} = J^{\Omega^{\xi}}((f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega} - \eta(f_{i-\varepsilon} - (f_i \bullet T_i)^{\Omega})) \tag{5.10}$$

$J^{i\xi}$ is the pseudo inverse of J^i, and the Newton step $\triangle t^\Omega$ updates T_i^Ω iteratively, as:

$$T_i^{\Omega+1} = T_i^\Omega + \triangle t_i^\Omega \tag{5.11}$$

Closed-Form Solution of $vec(o_i)$

To obtain the closed-form solution, the partial derivative over $vec(o_i)$ is calculated, as follows:

$$\sum_{j=1}^{k}(sign(vec(\overline{o}_{i,j}))) - \frac{\lambda}{2}\sum_{j=1}^{k}(\overline{\mathcal{N}}_{i-\varepsilon,j}^{\top} vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j})) = 0 \tag{5.12}$$

where $\tilde{\overline{f}}_{i,j}$ represents $\overline{f}_{i,j}$ aligned by T_i^*. The former equation is further expanded as:

$$\begin{aligned}(sign(vec(\overline{o}_{i,1})) - \tfrac{\lambda}{2}\overline{\mathcal{N}}_{i-\varepsilon,1}^{\top} vec(\tilde{\overline{f}}_{i,1} - \overline{o}_{i,1})) + \dots \\ +(sign(vec(\overline{o}_{i,k})) - \tfrac{\lambda}{2}\overline{\mathcal{N}}_{i-\varepsilon,k}^{\top} vec(\tilde{\overline{f}}_{i,k} - \overline{o}_{i,k})) = 0\end{aligned} \tag{5.13}$$

The optimal minimum value of o_i is obtained by minimizing each part of this expansion, e.g. $(sign(vec(\overline{o}_{t,1}))) - \frac{\lambda}{2}\mathcal{N}_{i-\varepsilon,j} vec(\tilde{\overline{f}}_{i,1} - \overline{o}_{t,1}))$.

By applying convex optimization theory [90] and the subdifferential ∂ [104] of each part of this expansion, the closed form solution to minimize $vec(\overline{o}_{i,j})$ is derived as follows:

$$0 = sign(vec(\overline{o}_{i,j})) - \frac{\lambda}{2}(\mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j})) \tag{5.14}$$

$$0 = \frac{2}{\lambda} sign(vec(\overline{o}_{i,j})) - \mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j}) + \mathcal{N}^{\top} vec(\overline{o}_{i,j}) \tag{5.15}$$

$$\mathcal{N}^{\top} vec(\overline{o}_{i,j}) = \frac{2}{\lambda} sign(vec(o)) + \mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j}) \tag{5.16}$$

$$\mathcal{N}^{\top} vec(\overline{o}_{i,j}) = \mathcal{S}_{\frac{2}{\lambda}}(\mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j})) \tag{5.17}$$

$$vec(\overline{o}_{i,j}) = \overline{\mathcal{N}}^{\top\xi}_{i-\varepsilon,j}(\mathcal{S}_{\frac{2}{\lambda}}(\overline{\mathcal{N}}_{i-\varepsilon,j} vec(\tilde{\overline{f}}_{i,j})) \tag{5.18}$$

where $\overline{\mathcal{N}}^{\top\xi}_{i-\varepsilon,j}$ is the pseudo inverse of $\overline{\mathcal{N}}_{i-\varepsilon,j}^{\top}$, and $\mathcal{S}_{\frac{2}{\lambda}}$ is the soft-thresholding operator

[51] The complete LNSP algorithm is shown in Algorithm 9.

Algorithm 9 Local Null Space Pursuit (LNSP)

Input: $f_{i-\varepsilon}$, $f_i, \triangle T_i^0$, γ, and α

Output: o_i

procedure :

  while Not Converge **do**

    $J = CalculateJacobian(f_{t-\varepsilon}, f_t \bullet T_i^{\Omega})$

    $\eta = \gamma(\frac{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega}\|}{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega-1}\|})^{\alpha}$

    $\triangle t^{\Omega} = J^{\Omega\xi}((f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega} - \eta(f_{i-\varepsilon} - (f_i \bullet T_i)^{\Omega}))$

    $T_i^{\Omega+1} = T_i^{\Omega} + \triangle t_i^{\Omega}$

  end while

  $\tilde{f}_i = f_i \bullet T_i$

  ForEach $vec(\tilde{\overline{f}}_{i,j}) \in vec(\tilde{f}_i)$

    $\overline{\mathcal{N}} = CalculateNullSpace(vec(\overline{b}_{i-\varepsilon,j})^{\top})$

    $vec(\overline{o}_{i,j}) = \overline{\mathcal{N}}^{\top\xi}_{i-\varepsilon,j}(\mathcal{S}_{\frac{2}{\lambda}}(\overline{\mathcal{N}}_{i-\varepsilon,j} vec(\tilde{\overline{f}}_{t,j})))$

  EndForEach

end procedure

5.4.3 Method Analysis

Although the LNSP method successfully formulates the PCP problem in a form that can be solved sequentially, it models the moving objects as only sparse (similar to I-MARO, N-MARO, and UT-MARO). Hence, there is no guarantee that o contains only truly moving objects.

5.5 KR-LNSP: Kinematic Regularization with Local Null Space Pursuit

The motivation of KR-LNSP method is to handle the drawbacks of the LNSP method and ensure that o contains only actual moving objects and, hence, reduce FPR. Similar to the KR-MARO method, the KR-LNSP method uses same kinematic regularization to model the moving objects as moving sparse. However, calculating the kinematic regularization term is time consuming due to the intensive searching for $CMOs$ and LMs in a frame. Therefore, KR-LNSP uses a cross search algorithm [105] to reduce the computational load.

5.5.1 Problem Formulation

To enhance the detection results, the KR-LNSP method is proposed. It integrates the kinematic regularization term into the LNSP problem formulation (i.e. 5.4) as follows:

$$\mathcal{L}(vec(o_i), vec(d_i)) = \sum_{j=1}^{k}(\|vec(\overline{\Gamma_{i,j}}\overline{o}_{i,j})\|_1) - \lambda\sum_{j=1}^{k}(\|\overline{\mathcal{N}}_{i-\varepsilon,j}^{\top}vec(\overline{f}_{i,j} - \overline{o}_{i,j} - \overline{d}_{i,j}) \bullet T_i)\|_2) \tag{5.19}$$

where $\overline{\Gamma_{i,j}}$ denotes the kinematic regularization term for $\overline{f}_{i,j}$ that penalizes the detected moving objects based on their motion uniqueness, and $\overline{s}_{i,j}$ signifies the false detections.

5.5.2 Problem Formulation Solution

Similar to LNSP, KR-LNSP calculate T using the inexact Newton method as shown in Eq. 5.11. The closed form solution of $\overline{o}_{i,j}$ is derived as follows:

$$0 = vec(\overline{\Gamma}_{i,j}\ Sign(\overline{o}_{i,j})) - \frac{\lambda}{2}(\mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j} - \overline{d}_{i,j})) \tag{5.20}$$

$$0 = \frac{2}{\lambda}vec(\overline{\Gamma}_{i,j}\ Sign(\overline{o}_{i,j})) - \mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{d}_{i,j}) + \mathcal{N}^\top vec(\overline{o}_{i,j}) \tag{5.21}$$

$$\mathcal{N}^\top \overline{o}_{i,j} = \frac{2}{\lambda}vec(\overline{\Gamma}_{i,j}\ Sign(\overline{o}_{i,j})) + \mathcal{N}^\top(\tilde{\overline{f}}_{i,j} - \overline{d}_{i,j}) \tag{5.22}$$

$$\mathcal{N}^\top \overline{o}_{i,j} = \mathcal{S}_{\frac{2}{\lambda}vec(\overline{\Gamma}_{i,j})}(\mathcal{N}^\top(\tilde{\overline{f}}_{i,j} - \overline{d}_{i,j})) \tag{5.23}$$

$$\overline{o}_{i,j} = \mathcal{N}^{\top\xi}(\mathcal{S}_{\frac{2}{\lambda}vec(\overline{\Gamma}_{i,j})}(\mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{d}_{i,j})) \tag{5.24}$$

The closed form solution of $\overline{d}_{i,j}$ is derived as follows

$$0 = \frac{\lambda}{2}(\mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j} - \overline{d}_{i,j})) \tag{5.25}$$

$$0 = \mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j} - \overline{d}_{i,j}) \tag{5.26}$$

$$\mathcal{N}^\top vec(\overline{d}_{i,j}) = \mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j}) \tag{5.27}$$

$$vec(\overline{d}_{i,j}) = \mathcal{N}^{\top\xi}(\mathcal{N}^\top vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j})) \tag{5.28}$$

Since calculating Γ is a computationally expensive process, especially when searching for CMO and LM, KR-LNSP uses the cross search algorithm [105] to speed up the searching for $CMO_{i,j}$ and $LM_{i,j,k}$. the cross search algorithm is a logarithmic step search where at each search step only 4 locations are tested, as shown Figure 5.3. First, a searching step (p) is defined, then cross correlation is applied at 4 locations that are far from the center of the searching window by p. At the best match location

half the search step $p/2$ and re-apply cross correlation at 4 locations that is far from the best matched location by $p/2$. Then repeat this process until the search step reached 1. In figure, 5.3 arrows 1, 2, and 3 show the direction of the best matched locations which are defined by solid red circles; the other non-solid circles show where cross correlation is applied at each search step. The complete algorithm of KR-LNSP is shown in Algorithm 10.

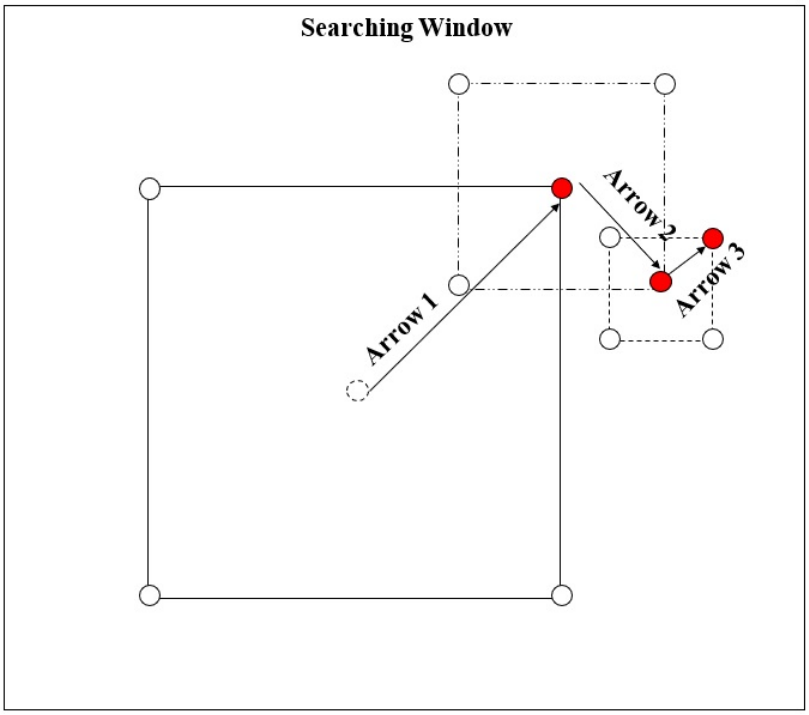


Figure 5.3: cross search algorithm

5.6 Results

For an objective evaluation of LNSP and KR-LNSP, they are compared with existing sequential-based and batch-based PCP methods using the same datasets and the computer specification described in section 3.7. The compared sequential PCP methods are: COROLA [106] , PracReProCS-pPCA [78], GOSUS [75], ROSETA [73], and

Algorithm 10 Kinematic Regularization with Local Null Space Pursuit (KR-LNSP)

Input: $f_{i-\varepsilon}$, $f_i, \triangle T_i^0$, γ, and α

Output: o_i

procedure :

  while Not Converge **do**

    $J = CalculateJacobian(f_{t-\varepsilon}, f_t \bullet T_i^{\Omega})$

    $\eta = \gamma(\frac{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega}\|}{\|(f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega-1}\|})^{\alpha}$

    $\triangle t^{\Omega} = J^{\Omega\xi}((f_{i-\varepsilon} - (f_i \bullet T_i))^{\Omega} - \eta(f_{i-\varepsilon} - (f_i \bullet T_i)^{\Omega}))$

    $T_i^{\Omega+1} = T_i^{\Omega} + \triangle t_i^{\Omega}$

  end while

  $\tilde{f}_i = f_i \bullet T_i$

  while Not Coverage **do**

    ForEach $vec(\tilde{\overline{f}}_{i,j}) \in vec(\tilde{f}_i)$

      $\overline{\mathcal{N}} = CalculateNullSpace(vec(\overline{b}_{i-\varepsilon,j})^{\top})$

      $vec(\overline{o}_{i,j}) = \mathcal{N}^{\top\xi}(\mathcal{S}_{\frac{2}{\lambda} vec(\overline{\Gamma}_{i,j})}(\mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j} - \overline{d}_{i,j}))$

      $vec(\overline{d}_{i,j}) = \mathcal{N}^{\top\xi}(\mathcal{N}^{\top} vec(\tilde{\overline{f}}_{i,j} - \overline{o}_{i,j}))$

    EndForEach

    $\Gamma_i = CalculateRegularizationTerm$ *//as shown in Algorithm 8*

  end while

end procedure

incPCP [80]. The compared batch-based PCP methods are KR-MARO, I-MARO, Spring-MARO, UT-MARO, N-MARO, RASL-IALM [65] , RASL-APG [63], DE-COLOR [54], and 3TD [55].

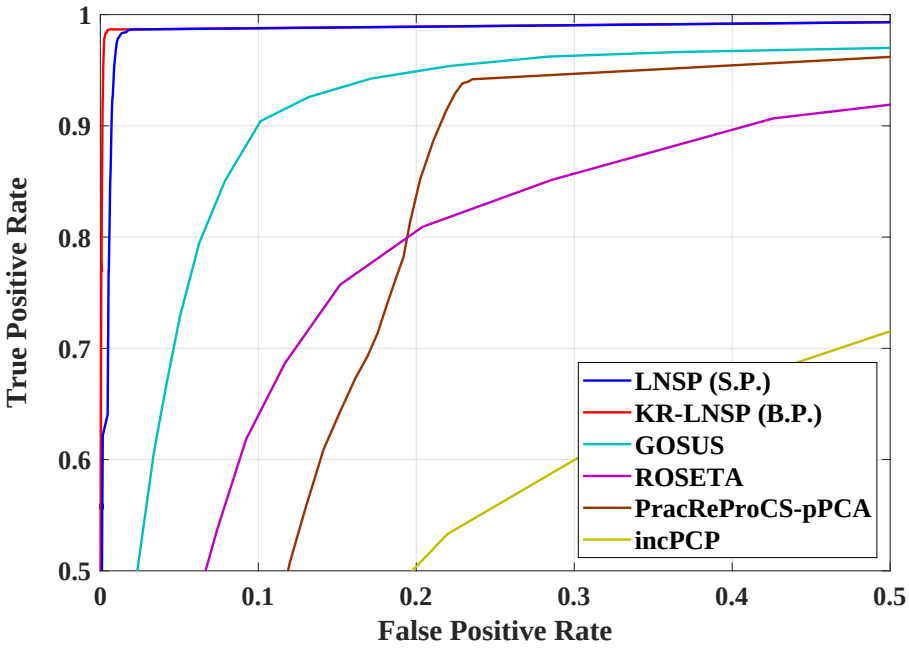


Figure 5.4: KR-LNSP versus LNSP versus sequential PCP detection methods

Accuracy

By analysing Figure 5.4, Figure 5.5, Figure 5.6, and Table 5.6, it is evident that KR-LNSP outperforms both sequential-based and batch-based PCP methods. KR-LNSP provides a high TPR of 98% on average with a significantly low FPR of 0.4% on average. Although LNSP and COROLA achieve high TPRs, their false detections are much higher compared to KR-LNSP (FPR is 3% and 9%, respectively). In PracReProCS-pPCA, ROSETA, and incPCP, the false detections are very high, e.g. their FPR is 19%, 42% and 39%, respectively. GOSUS does not have a high FPR as in PracReProCS-pPCA, but it sacrifices the TPR (90% on average). Most compared

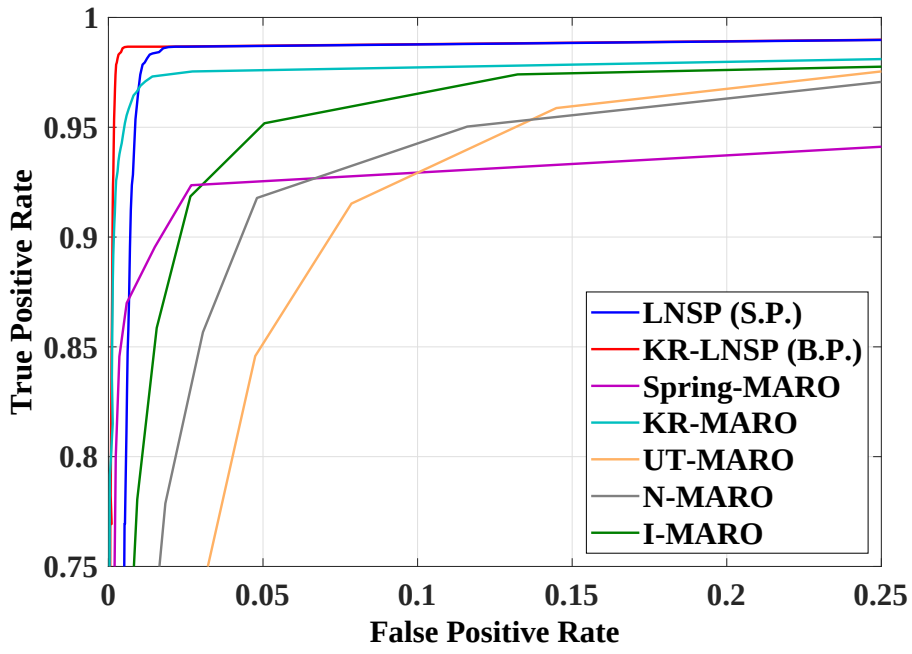


Figure 5.5: KR-LNSP versus LNSP versus our batch-based PCP detection methods

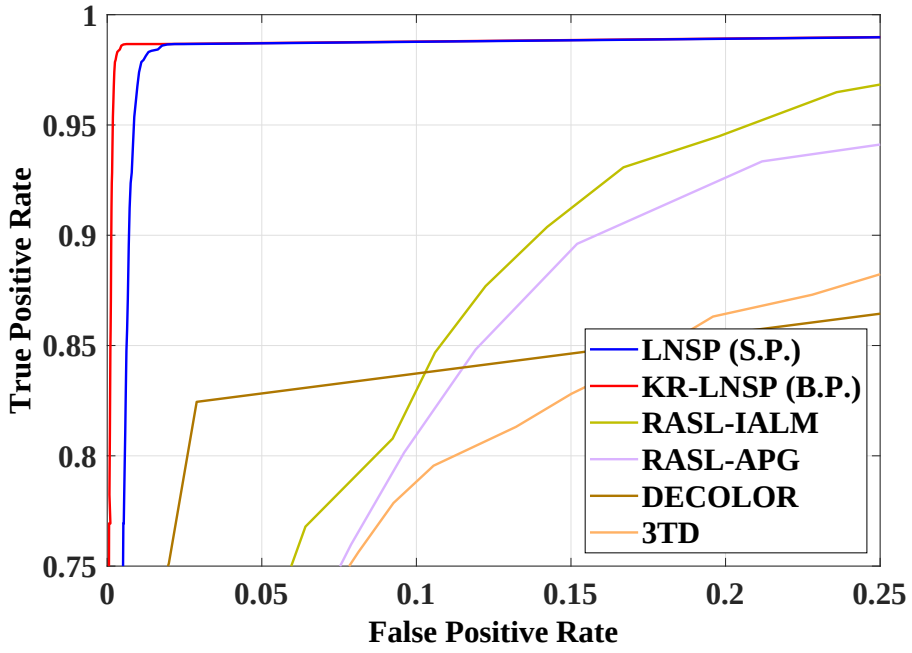


Figure 5.6: KR-LNSP versus LNSP versus other PCP detection methods

batch-based PCP methods have either a high TPR with a high FPR (e.g. I-MARO, RASL-MARO, N-MARO, and UT-MARO) or low FPR with low TPR (e.g. Spring-MARO and DECOLOR). For 3TD, it has a low TPR and a high FPR. The detailed TPR and FPR comparison between KR-LNSP, LNSP and other PCP methods (sequential and batch-based PCP detection methods) is found in Table 5.2 to Table 5.5. For sample results, please refer to figures 5.7 to 5.15. For the full video sequence, please visit our website: https://phdthesisvisualresults.weebly.com/

Execution time

As shown in Table 5.6 ROSETA has the best execution time compared to other methods, it takes 0.05 seconds per frame. KR-LNSP and LNSP are the second best methods in terms of the execution time, e.g. their execution time is 0.3 and 0.12 second per frame, respectively. incPCP and COROLA come in third place, they require around 1 second per frame. UT-MARO and I-MARO take 1.7 and 2.5 seconds on average to process a frame. Other methods in the comparison, such as PracReProCS-pPCA, GOSUS, RASL-IALM, N-MARO, Spring-MARO, DECOLOR, and 3TD, are computationally intensive and take longer times

By looking at the combined performance and execution time, Table 5.6, KR-LNSP provides the best accuracy compared to other PCP methods, either sequential-based or batch-based. However, KR-LNSP consumes more time per frame compared to LNSP. This time is consumed in calculating Γ which guarantees the significantly low FPR. On the opposite side, ROSETA achieves real-time performance, but its accuracy is significantly lower. The methods COROLA, GOSUS, KR-MARO, I-MARO, N-MARO, UT-MARO, and RASL-IALM have moderate accuracy (still lower than KR-LNSP), but they are computationally expensive. Spring-MARO, DECOLOR, and 3TD suffer from a low accuracy and quite high computational loads. incPCP is not

Table 5.1: KR-LNSP versus LNSP versus current state-of-the-art methods

Type	Method	Average TPR	Average FPR	Execution time (seconds per frame)	Implementation
Sequential PCP Methods	LNSP	**98%**	3%	**0.1**	Matlab & C++
	KR-LNSP	**98%**	**0.4%**	0.3	Matlab & C++
	COROLA	96%	9%	0.9	Matlab & C++
	PracRePro	93%	19%	14	Matlab
	CS-pPCA	90%	8%	21	Matlab
	GOSUS	90%	42%	**0.05**	Matlab
	ROSETA	71.5%	39%	0.8	Matlab
	incPCP	89%	3%	6	Matlab
Batch-based PCP Methods	Spring-MARO	97%	**0.4%**	10	Matlab
	KR-MARO	91%	10%	1.7	Matlab
	UT-MARO	93%	9%	5.8	Matlab
	N-MARO	95.5%	4.5%	2.5	Matlab
	I-MARO	93.5%	11%	4.5	Matlab
	RASL-IALM	90.3%	15.3%	60	Matlab
	RASL-APG	79%	6%	6.2	Matlab
	DECOLOR	86.4%	18.2%	10.1	Matlab
	3TD				

Table 5.2: Complete TPR and FPR comparison between LNSP and sequential PCP detection methods

Dataset		LNSP		COROLA[1]		PracReProCS-pPCA		GOSUS		ROSETA		incPCP	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest1	98%	**1.4%**	98%	4.8%	**99%**	19%	98%	9%	98%	20%	80.2%	21%
	Egtest2	97%	**0.6%**	98%	4%	**100%**	23%	83%	19%	**100%**	27%	81%	27%
	Egtest3	**98%**	**0.4%**	94%	7%	97%	10%	94%	10%	97%	40%	85.2%	35%
	Egtest5	**98%**	**0.9%**	93%	5%	94%	22%	92%	3%	97%	40%	62.4%	35%
UCF	Action1	**99%**	**6%**	98%	11%	77%	18%	85%	8%	83%	41%	30%	21%
	Action2	97%	**8%**	**98%**	22%	**98%**	20%	96%	9%	91%	37%	60%	25%
	Action3	**96%**	**0.5%**	95%	8%	94%	20%	83%	9%	42%	29%	56%	26%
VIRAT	flight2tape1_2	**100%**	**2%**	–	–	90%	33%	85%	6%	**100%**	77%	88%	50%
	flight2tape2_1	**100%**	**5%**	–	–	91%	34%	93%	13%	99%	73%	84%	70%
	flight2tape3_2	98%	**3%**	–	–	**100%**	31%	98%	1%	**100%**	68%	89%	80%

[1] COROLA is not tested by sequences flight2tap1_2, flight2tap2_1 and flight2tap3_2 as the provided online COROLA implementation crashes when using these sequences

Table 5.3: Complete TPR and FPR comparison between KR-LNSP and sequential PCP detection methods

Dataset		KR-LNSP		COROLA		PracReProCS-pPCA		GOSUS		ROSETA		incPCP	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest1	98%	**0.5%**	98%	4.8%	**99%**	19%	98%	9%	98%	20%	80.2%	21%
	Egtest2	97%	**0.2%**	98%	4%	**100%**	23%	83%	19%	**100%**	27%	81%	27%
	Egtest3	**98%**	**0.2%**	94%	7%	97%	10%	94%	10%	97%	40%	85.2%	35%
	Egtest5	**98%**	**0.2%**	93%	5%	94%	22%	92%	3%	97%	40%	62.4%	35%
UCF	Action1	**99%**	**0.4%**	98%	11%	77%	18%	85%	8%	83%	41%	30%	21%
	Action2	97%	**0.2%**	**98%**	22%	**98%**	20%	96%	9%	91%	37%	60%	25%
	Action3	**96%**	**0.1%**	95%	8%	94%	20%	83%	9%	42%	29%	56%	26%
VIRAT	flight2tape1_2	**100%**	**0.1%**	–	–	90%	33%	85%	6%	**100%**	77%	88%	50%
	flight2tape2_1	**100%**	**2%**	–	–	91%	34%	93%	13%	99%	73%	84%	70%
	flight2tape3_2	98%	**0.1%**	–	–	**100%**	31%	98%	1%	**100%**	68%	89%	80%

Table 5.4: Complete TPR and FPR comparison between KR-LNSP and our batch-based PCP detection methods

Dataset		**KR-LNSP**		**Spring-MARO**		**KR-MARO**		**UT-MARO**		**N-MARO**		**I-MARO**	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest1	**98%**	**0.5%**	97%	0.09%	**98%**	1%	97%	1%	96%	0.3%	96%	0.3%
	Egtest2	97%	**0.2%**	**100%**	0.3%	99%	**0.2%**	97%	0.1%	99%	0.7%	99%	0.7%
	Egtest3	**98%**	**0.2%**	98%	0.7%	**98%**	**0.2%**	**98%**	0.9%	97%	0.9%	97%	0.9%
	Egtest5	**98%**	**0.2%**	93%	0.9%	92%	0.4%	91%	0.6%	90%	1%	90%	1%
UCF	Action1	**99%**	**0.4%**	87%	3%	98%	**0.4%**	93%	4%	86%	10%	95%	10%
	Action2	97%	**0.2%**	94%	4%	97%	**0.2%**	81%	4%	96%	7%	**98%**	4%
	Action3	**96%**	**0.1%**	60%	1%	**96%**	0.3%	95%	5%	90%	7%	**96%**	5%
VIRAT	flight2tape1_2	**100%**	**0.1%**	91%	2%	98%	**0.1%**	80%	40%	94%	9%	95%	4%
	flight2tape2_1	**100%**	**2%**	85%	10%	90%	**2%**	90%	25%	91%	32%	92%	10%
	flight2tape3_2	**98%**	0.1%	94%	4%	**98%**	**0.1%**	90%	10%	94%	9%	96%	8%

Table 5.5: Complete TPR and FPR comparison between KR-LNSP and other batch-based PCP detection methods

	Dataset	KR-LNSP		RASL-IALM		RASL-APG		DECOLOR		3TD	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
DARPA	Egtest1	**98%**	**0.5%**	95%	6%	95%	7%	92%	1%	95%	5%
	Egtest2	97%	**0.2%**	**98%**	6%	94%	13%	92%	1.5%	93%	12%
	Egtest3	**98%**	**0.2%**	95%	5%	87%	4%	84%	2%	84%	8%
	Egtest5	**98%**	**0.2%**	88%	7%	94%	2%	85%	1.7%	87%	20%
UCF	Action1	**99%**	**0.4%**	94%	17%	92%	20%	72%	11%	85%	20%
	Action2	**97%**	**0.2%**	95%	13%	92%	11%	71%	7%	85%	15%
	Action3	**96%**	**0.1%**	92%	10%	91%	12%	68%	5%	70%	10%
VIRAT	flight2tape1_2	**100%**	**0.1%**	94%	9%	80%	40%	70%	5%	78%	35%
	flight2tape2_1	**100%**	**2%**	91%	23%	90%	25%	80%	12%	93%	30%
	flight2tape3_2	**98%**	**0.1%**	94%	9%	90%	10%	85%	9%	90%	20%

very computationally expensive, but its accuracy is the worst.

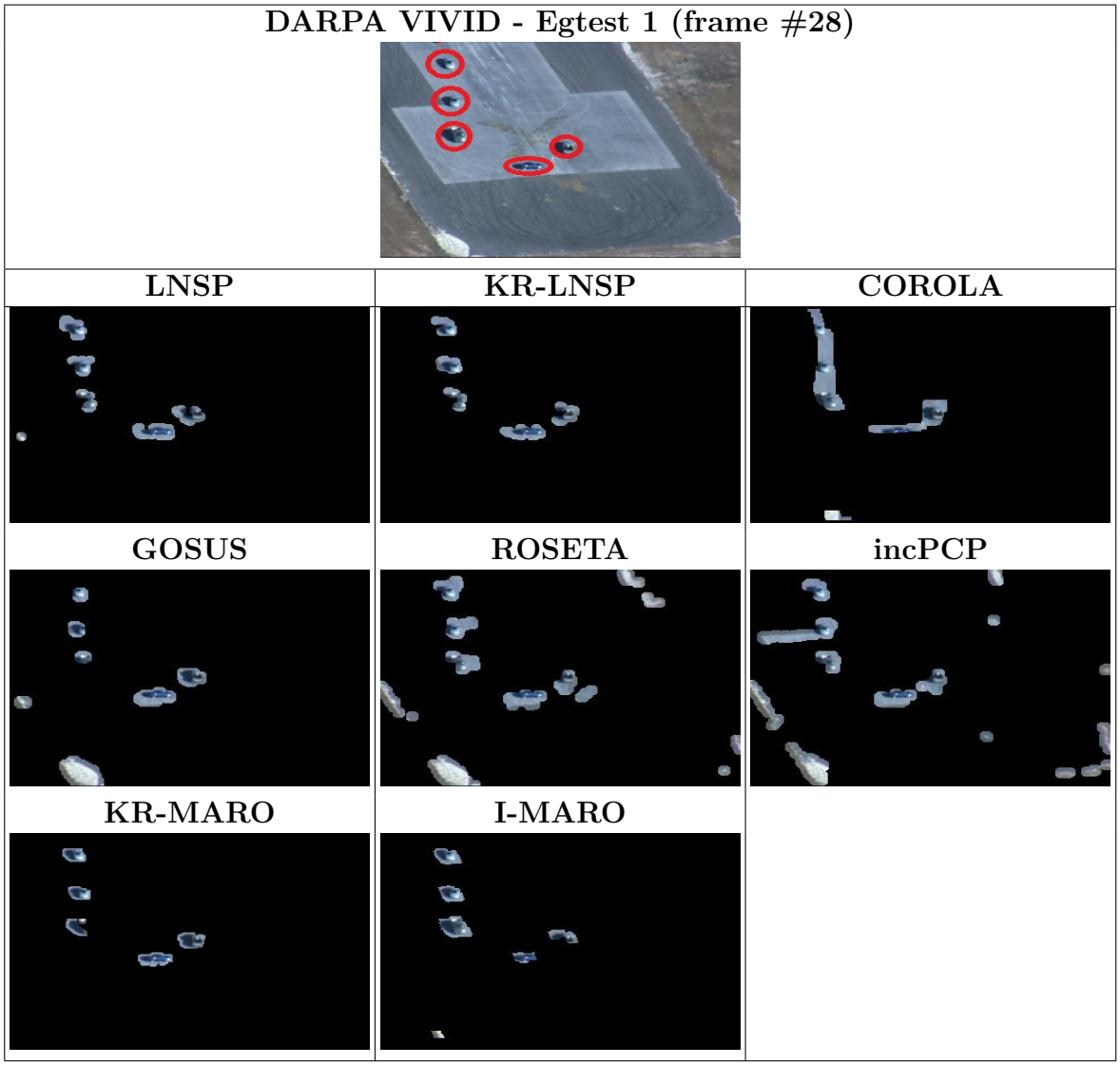


Figure 5.7: Sample results of the compared PCP detection methods on DARPA VIVID

5.7 Conclusion

This chapter presents two novel sequential PCP methods, namely LNSP and KR-LNSP. These methods successfully outperform current state-of the-art methods at

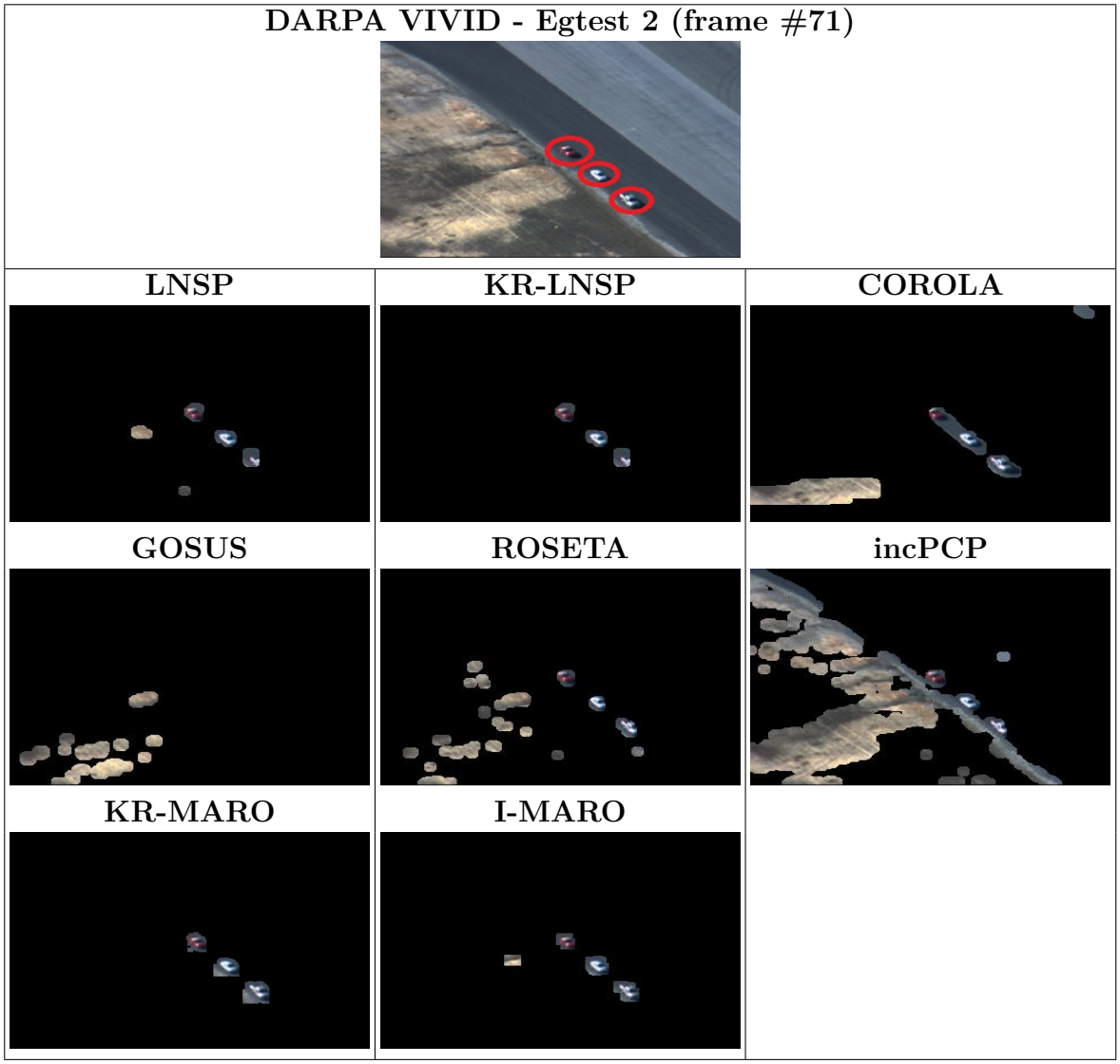


Figure 5.8: Sample results of the compared PCP detection methods on DARPA VIVID continued

both levels of detection accuracy (High TPR and low FPR) and reduce the computational loads. LNSP and KR-LNSP model the background as a subspace which lies in a low-dimension subspace and the moving objects as sparse corrupting a video. Based on the former modelling, LNSP and KR-LNSP propose a problem formulation that uses the multiple local null spaces to detect the moving objects. Although LNSP and KR-LNSP have similar values at TPR, KR-LNSP radically reduces the TPR,

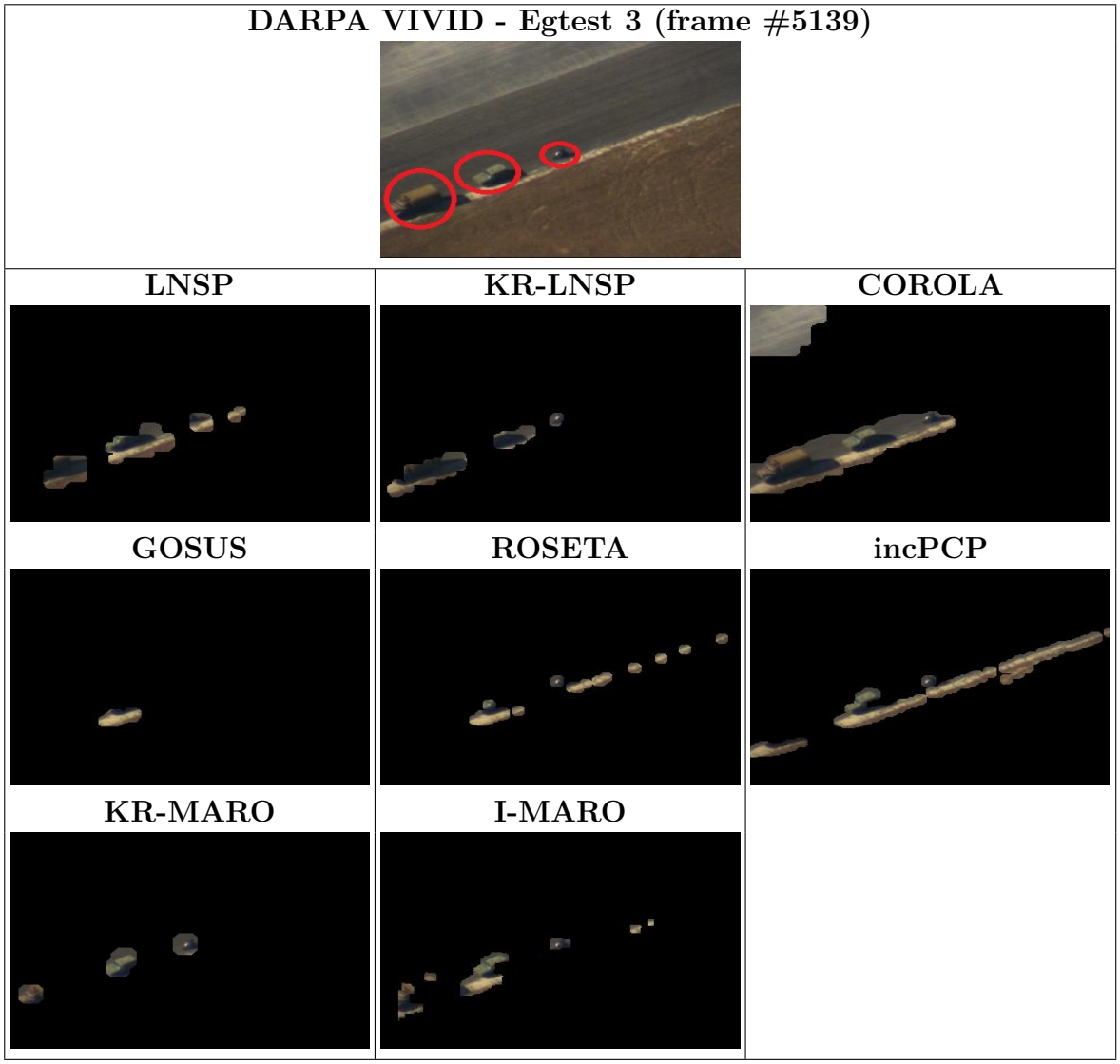


Figure 5.9: Sample results of the compared PCP detection methods on DARPA VIVID continued

compared to LNSP, thanks to the integration of the kinematic regularization term in the problem formulation.

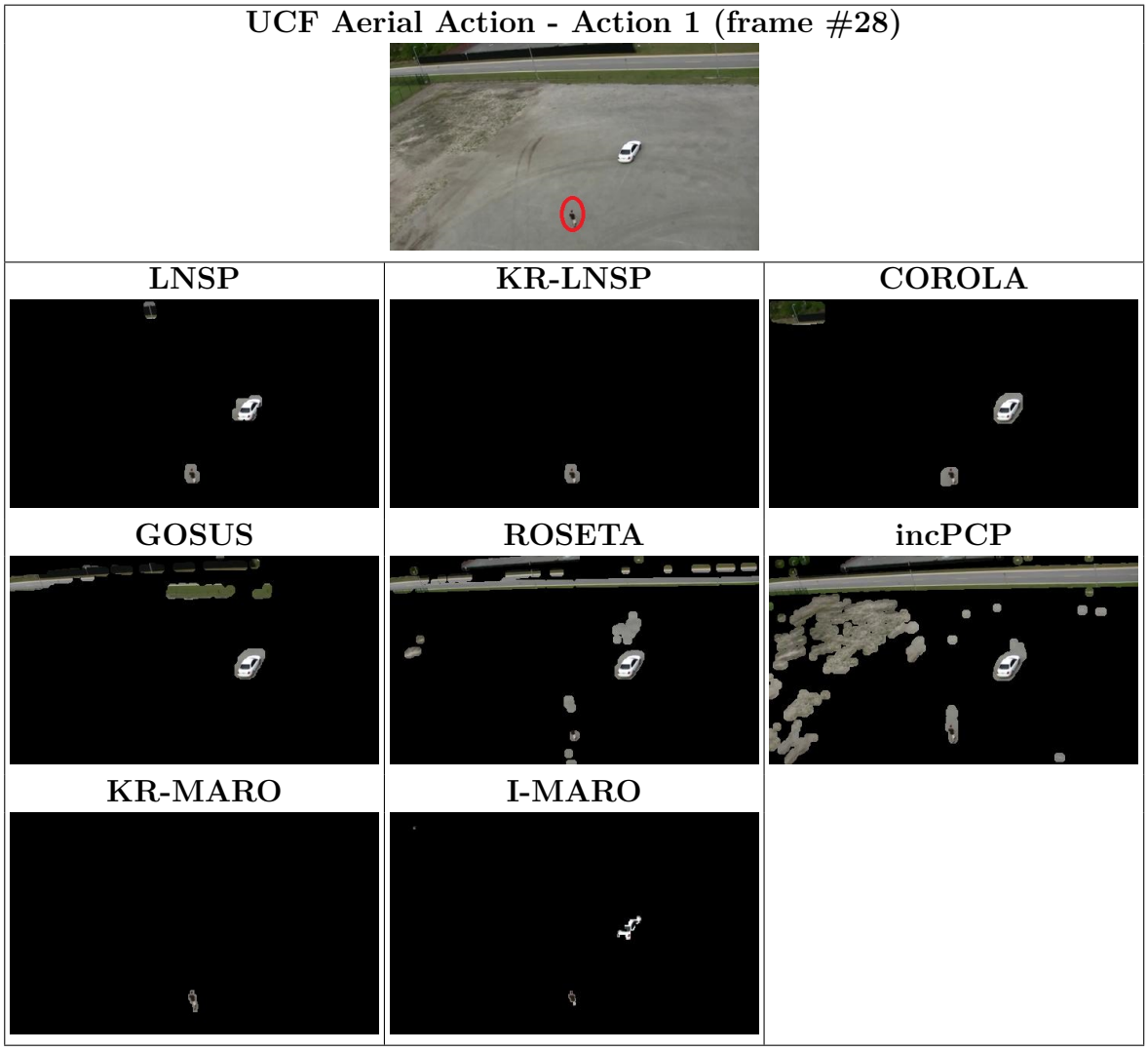


Figure 5.10: Sample results of the compared PCP detection methods on UCF aerial action

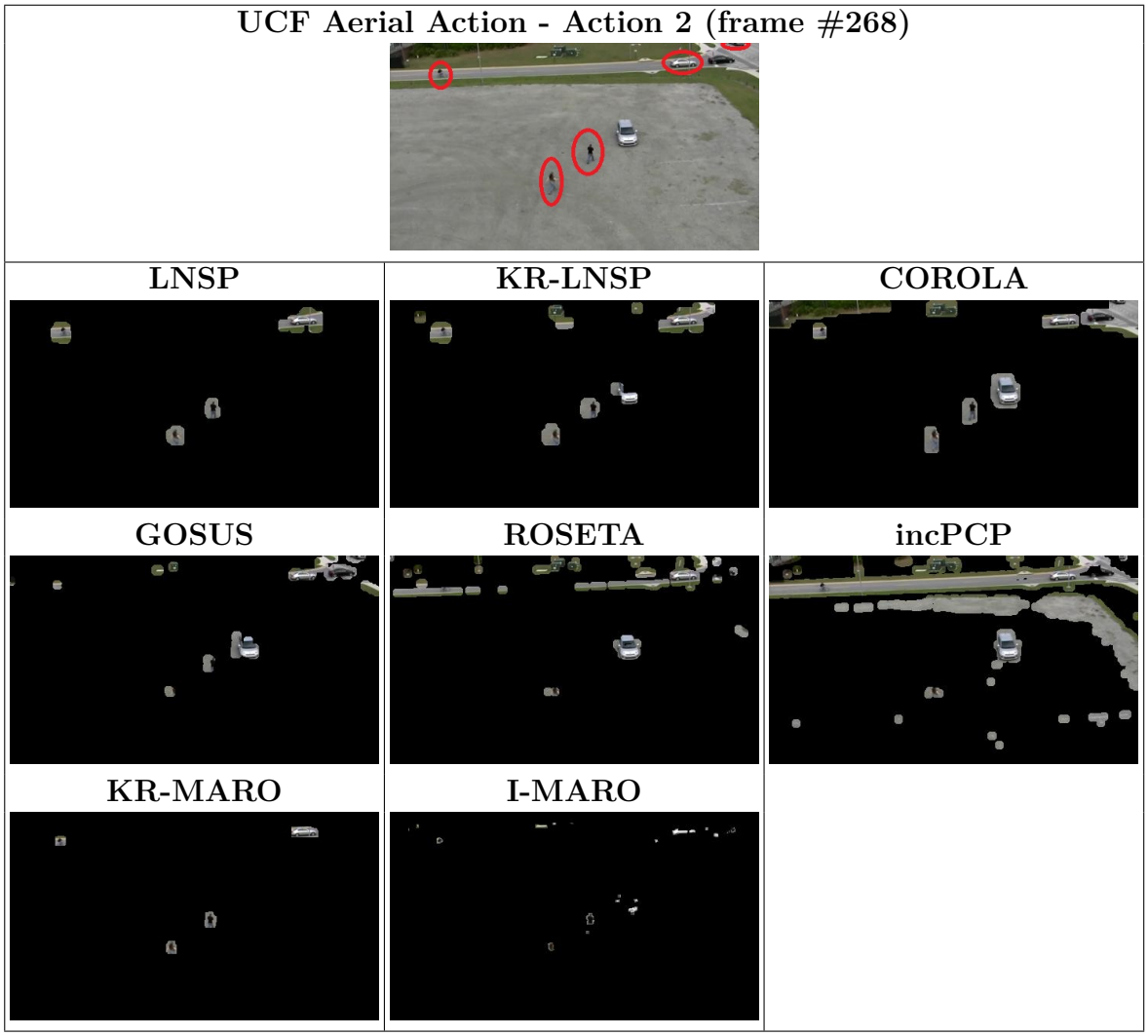


Figure 5.11: Sample results of the compared PCP detection methods on UCF aerial action continued

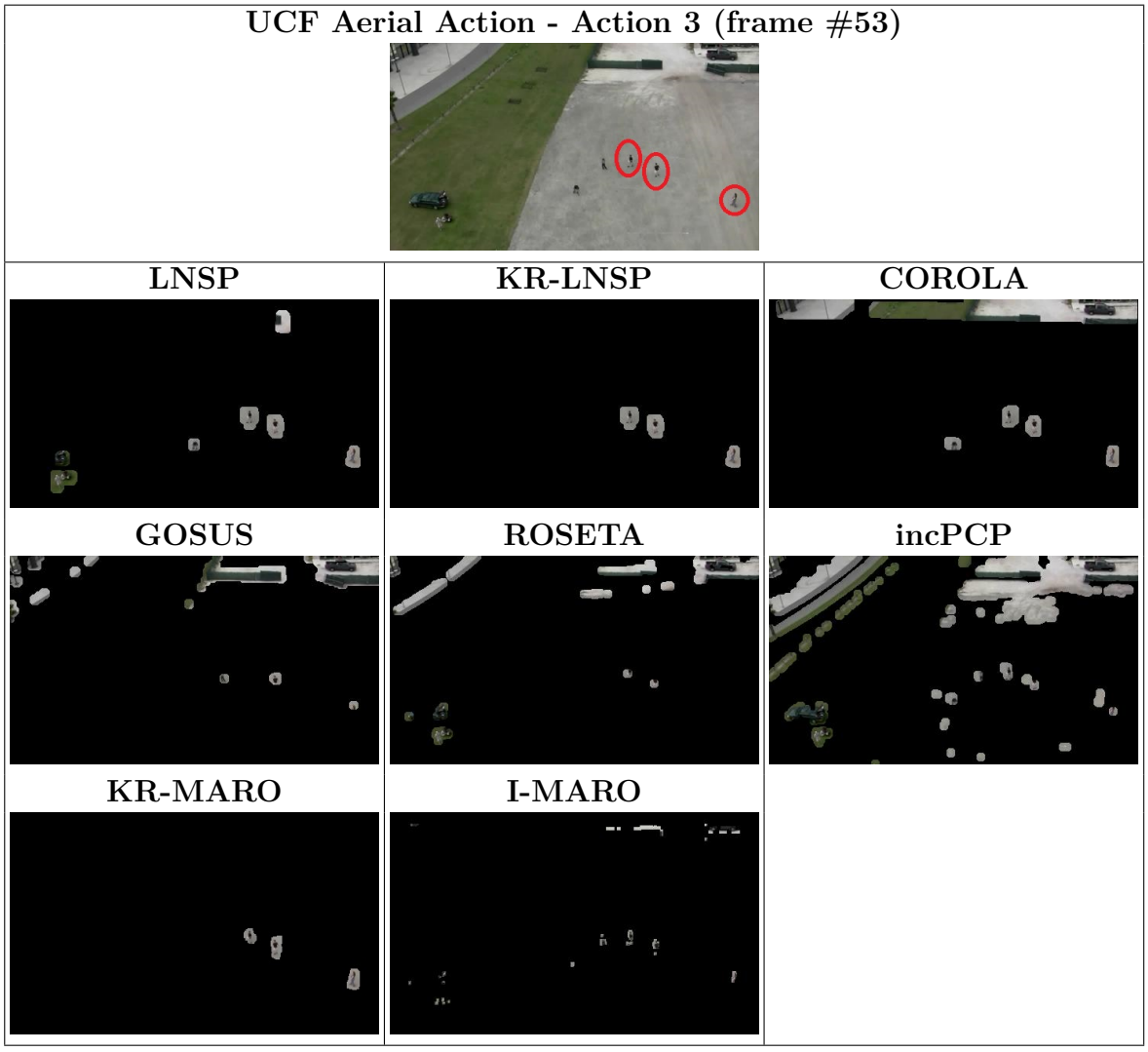


Figure 5.12: Sample results of the compared PCP detection methods on UCF aerial action continued

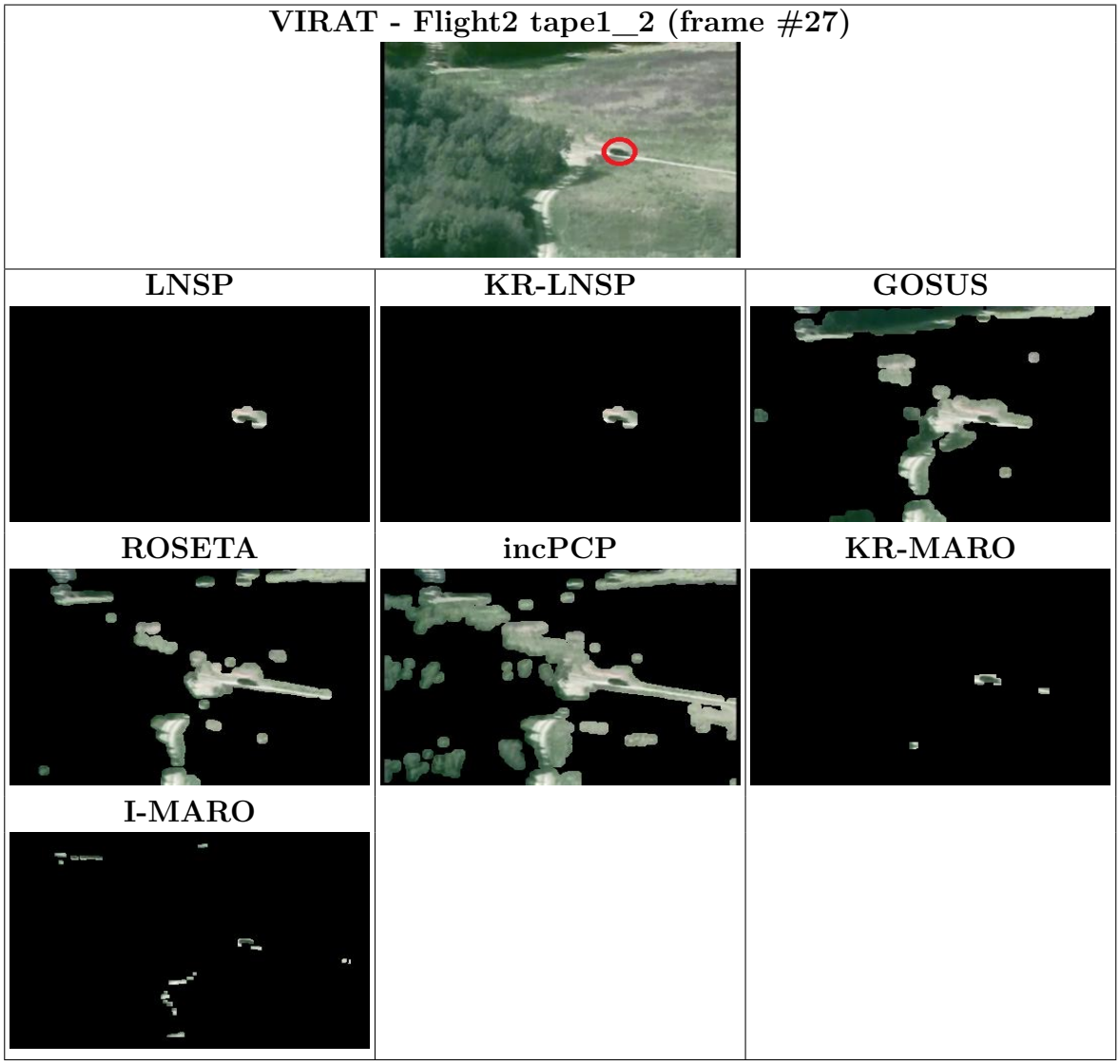


Figure 5.13: Sample results of the compared PCP detection methods on VIRAT

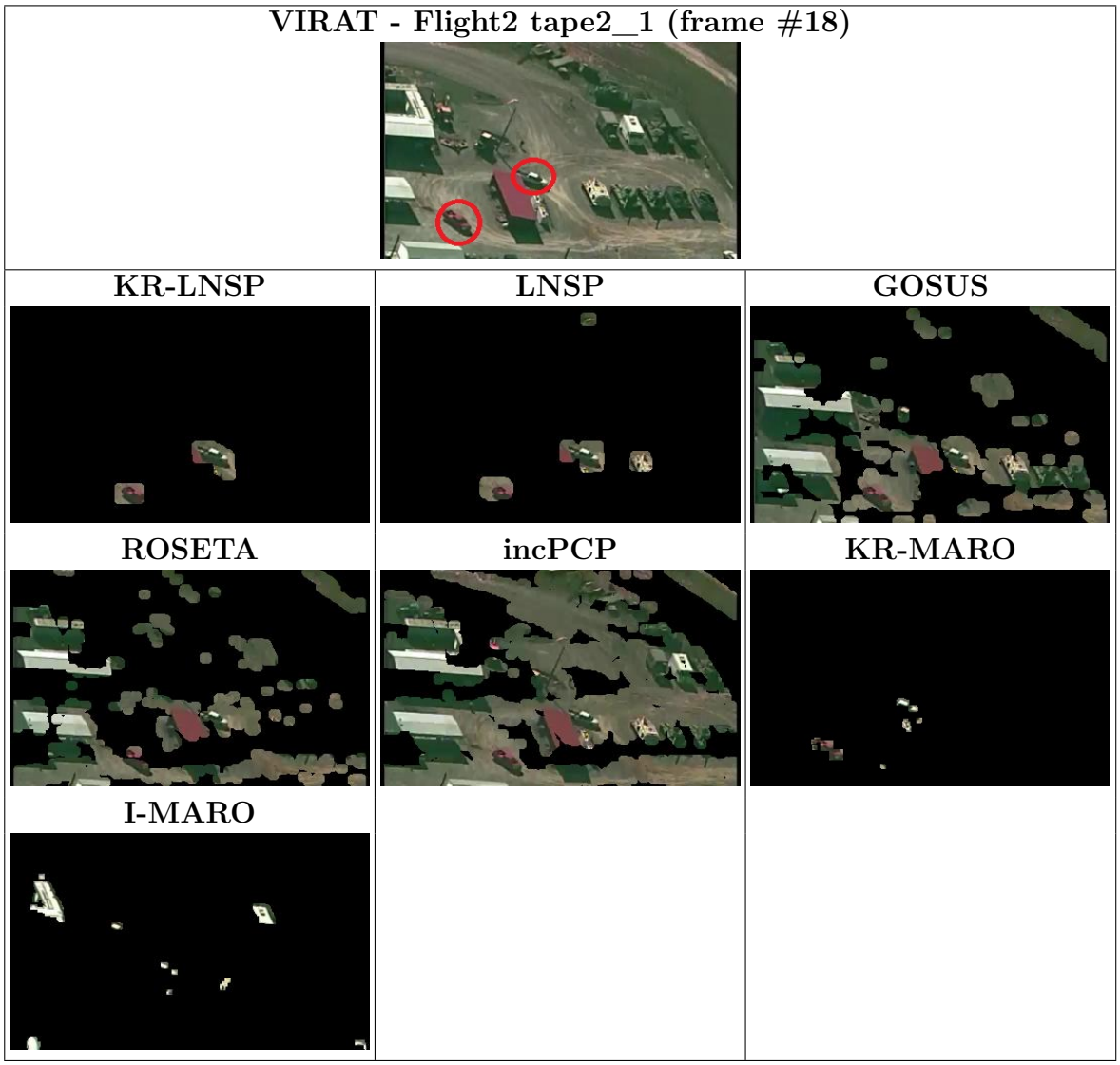


Figure 5.14: Sample results of the compared PCP detection methods on VIRAT continued

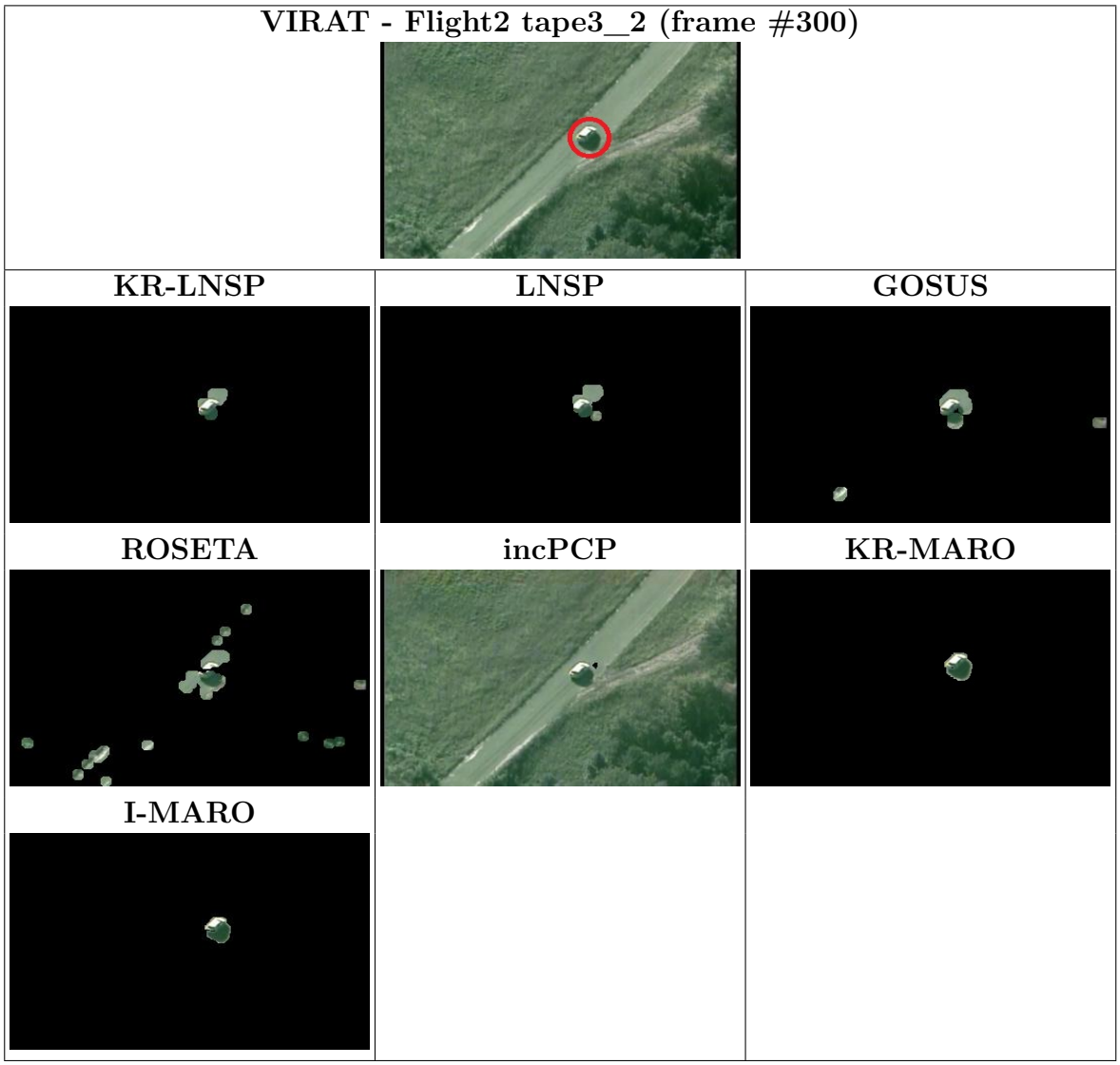


Figure 5.15: Sample results of the compared PCP detection methods on VIRAT continued

Chapter 6

Conclusion and Future Work

6.1 Thesis summary

This thesis studies the detection GMOs from aerial images. The motivation behind this thesis is the absence of robust GMOs detection method which is a key process of ACPs applications. Current-state-of-the-art methods suffer from at least one of the following: (1) high false detection rates, (2) low true detection rates, or (3) high computational loads. This is justified as current-state-of-the-art methods may have a restriction on the size of the moving objects, the shakiness of the ACPs, or both. Moreover, they depend on computationally expensive techniques to sense the motion of GMOs.

Therefore, the goal of this thesis is to provide a robust detection method that is characterized by a precise detection of small size moving objects (i.e., high TPR and low FPR), adaptability to high shaky motion of UAVs, and real-time performance.

To this end, the research in this thesis is demonstrated via seven novel detection methods: UT-MARO, N-MARO, I-MARO, Spring-MARO, KR-MARO, LNSP, and KR-LNSP. The first three methods, i.e. UT-MARO, N-MARO, and I-MARO, focus

on enhancing the frame alignment. UT-MARO uses the unscented transformation to estimate a transformation domain that aligns consecutive frames. N-MARO calculates the transformation domain using weighted-Newton method. I-MARO computes the transformation domain using inexact Newton method to balance between the accurate estimation of the transformation domain and the processing load.

Spring-MARO and KR-MARO are proposed to lower the false detection rates. They introduce a new problem formulation that models the true moving objects as moving sparse. Typically, a regularization term that encodes temporal information of the moving objects, is integrated into PCP formulation. The regularization term in Spring-MARO is based on a spring-model that considers transnational motion. KR-MARO proposes a kinematic regularization term based on an advanced system of springs to accurately reflect the kinematic properties of the moving objects.

LNSP and KR-LNSP methods extend the concept in PCP to achieve real-time performance. To this end, LNSP and KR-LNSP follow two strategies:

- Instead of modelling the background as a low-rank matrix, LNSP and KR-LNSP model the background as a subspace which lies on a low-dimensional subspace. Therefore, they can use the null space of the background in previous frames to detect the moving objects in an input frame; hence, the detection can be performed in a sequential manner.

- LNSP and KR-LNSP use multiple local null spaces to allow for real-time performance.

KR-LNSP proposes an extra step over LNSP, which is integrating the kinematic regularization term into the problem formulation to differentiate between truly moving objects as false detections. Consequently, KR-LNSP successfully balances between three attributes: 1) very low FPR via using the kinematic regularization, 2) high

TPR due to extending PCP concept via modelling the background as a subspace that lies on a low-dimension subspace, and 3) low computational load by using the inexact Newton method, local null spaces, and the fast template matching method to obtain the regularization term.

6.2 Achievements summary

Tables 6.1 and 6.2 summarize our achievements in comparison with existing PCP detection methods. First, we introduce two alignments methods based on the unscented transformation and inexact Newton method (UT-MARO and I-MARO). Second, the solution of the PCP problem is extended by using the backtracking behaviour (as shown in I-MARO and KR-MARO). Third, a regularization term is added to the PCP formulation that captures the motion cue of the moving objects (as in KR-MARO, Spring-MARO, and KR-LNSP). Finally, local null space is used to sequentially recover the background from input frames and reach real-time performance (as shown in LNSP and KR-LNSP).

From Table 6.1, our batch-based proposed methods (UT-MARO, N-MARO, I-MARO, and KR-MARO) outperform relative current state-of-the-art methods (RASL-IALM, RASL-APG, DECOLOR, and 3TD). However, their batch-based behaviour prevents them from reaching the real-time performance. Therefore, we propose sequential PCP method (LNSP) that achieves the real-time performance. From Table 6.1 and Table 6.2, our proposed sequential methods, i.e. LNSP and KR-LNSP, outperform both batch-based PCP methods and sequential PCP methods, with minimal processing time.

Table 6.1: Comparison summary between the proposed batch-based PCP methods and relevant current state-of-the-art methods

Method	**Problem Formulation**		**Problem Solution**		**Performance**		
	Moving Objects Modelling	**Penalizing Term**	**Video Decomposition**	**Transformation Domain Calculation**	**TPR**	**FPR**	**Execution time (seconds per frame)**
Spring-MARO	Moving Sparse	Spring Model	IALM	Weighted-Newton Method	88.9%	2.8%	6
KR-MARO	Moving Sparse	Kinematic regularization Term	IALM with Backtracking	Inexact Newton method	**97%**	**0.4%**	10
UT-MARO	Sparse	—	IALM	Unscented Transformation	90.6%	9.9%	**1.7**
N-MARO	Sparse	—	IALM	Weighted-Newton Method	93%	8.8%	5.8
I-MARO	Sparse	—	IALM with Backtracking	Inexact Newton Method	95.4%	4.8%	2.5
RASL-IALM	Sparse	—	IALM	Newton Method	93.5%	11%	4.5
RASL-APG	Sparse	—	APG	Newton method	90.3%	15.3%	60
DECOLOR	Sparse	—	SOFT-IMPUTE	Weighted-Newton Method	79%	6%	6.2
3TD	Sparse with Unique Optical Flow Characteristics	Object Confidence Map	IALM	—	86.4%	18.2%	10.1

6.3 Future directions of the thesis

The growing of population of vehicles has an impact on computer vision processes. One of these processes that has been affected is detecting moving objects. Typically, the scenes in most urban areas are overcrowded with pedestrians, vehicles, bikes, etc.

Table 6.2: Comparison summary between the proposed sequential PCP methods and relevant current state-of-the-art methods

Method	**Problem Formulation**		**Problem Solution**		**Performance**		
	Moving Objects Modelling	**Subspace**	**Video Decomposition**	**Handling Camera Motion**	**TPR**	**FPR**	**Execution time (seconds per frame)**
KR-LNSP	Moving Sparse with Regularization Term	Local Null Space	Lagrange Multiplier	Calculate Transformation Domain Using Inexact Newton Method	98%	0.4%	0.3
LNSP	Sparse	Local Null Space	Lagrange Multiplier	Calculate Transformation Domain Using Inexact Newton Method	98%	3%	0.12
COROLA	Sparse with spatial connectively	Matrix Basis	OR-PCA	Calculate Transformation Domain Using Weighted-Newton Method	96%	9%	0.9
PracReProCS-pPCA	Sparse	Null Space	Lagrange Multiplier	Updating Subspace	93%	19%	14
GOSUS	Sparse with Spatial Connectively	Matrix Basis	Lagrange Multiplier	Updating Subspace	90%	8%	21
ROSETA	Sparse	Matrix Basis	Lagrange Multiplier	Updating Subspace	90%	42%	**0.05**
incPCP	Sparse	Matrix Basis	ThinSVD	Updating Subspace	71.1%	39%	0.8

Since, our future is oriented towards using ACPs, especially UAVs, so the detection of GMOs should adapt to such overcrowded scenes. In other words, the new challenge that should be addressed is the ability to detect small size objects where intermediate spaces between these objects are extremely small. Therefore, the proposed kinematic regularization term has to be calculated using a network of springs that connect candidate moving objects with each other and group of these springs connected to a static landmark. In that case, the forces of these springs can be used to determine the true moving objects.

References

[1] A. Ryan and J. K. Hedrick, "A mode-switching path planner for uav-assisted search and rescue," in *Decision and Control, 2005 and 2005 European Control Conf. CDC-ECC'05. 44th IEEE Conf on.* IEEE, 2005, pp. 1471–1476.

[2] W.-L. Yang, L. Lei, and J.-S. Deng, "Optimization and improvement for multi-uav cooperative reconnaissance mission planning problem," in *ICCWAMTIP, 2014 11th Int Computer Conf on.* IEEE, 2014, pp. 10–15.

[3] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.

[4] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970–2978, 2014.

[5] K. A. Joshi and D. G. Thakore, "A survey on moving object detection and tracking in video surveillance system," *IJSCE, ISSN*, pp. 2231–2307, 2012.

[6] S. Y. Elhabian, K. M. El-Sayed, and S. H. Ahmed, "Moving object detection in spatial domain using background removal techniques-state-of-art," *Recent patents on computer science*, vol. 1, no. 1, pp. 32–54, 2008.

[7] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 5–16, 1994.

[8] S. Ali and M. Shah, "Cocoa: tracking in aerial imagery," in *Defense and Security Symposium*. Intern Society for Optics and Photonics, 2006, pp. 62 090D–62 090D.

[9] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations," *Circuits and Systems for Video Technology, IEEE Trans on*, vol. 4, no. 3, pp. 339–356, 1994.

[10] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 7, pp. 780–785, 1997.

[11] E. Hayman and J.-O. Eklundh, "Statistical background subtraction for a mobile observer," in *null*. IEEE, 2003, p. 67.

[12] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-based surveillance systems*. Springer, 2002, pp. 135–144.

[13] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[14] K. Moo Yi, K. Yun, S. Wan Kim, H. Jin Chang, and J. Young Choi, "Detection of moving objects with non-stationary cameras in 5.8 ms: Bringing motion detection to your mobile device," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 27–34.

[15] A. Bevilacqua and P. Azzari, "High-quality real time motion detection using ptz cameras," in *AVSS'06. IEEE Int Conf on.* IEEE, 2006, pp. 23–23.

[16] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int j comput vision*, vol. 74, no. 1, pp. 59–73, 2007.

[17] A. Ollero, J. Ferruz, F. Caballero, S. Hurtado, and L. Merino, "Motion compensation and object detection for autonomous helicopter visual navigation in the comets system," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 19–24.

[18] J. Ferruz and A. Ollero, "Real-time feature matching in image sequences for non-structured environments. applications to vehicle guidance," *Journal of Intelligent & Robotic Systems*, vol. 28, no. 1, pp. 85–123, 2000.

[19] M. Siam, R. ElSayed, and M. ElHelw, "On-board multiple target detection and tracking on camera-equipped aerial vehicles," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on.* IEEE, 2012, pp. 2399–2405.

[20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[21] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and vision Computing*, vol. 15, no. 1, pp. 59–76, 1997.

[22] X. Li, M. Chen, F. Nie, and Q. Wang, "A multiview-based parameter free framework for group detection." in *AAAI*, 2017, pp. 4147–4153.

[23] W. Xu, S. Zhong, L. Yan, F. Wu, and W. Zhang, "Moving object detection in aerial infrared images with registration accuracy prediction and feature points selection," *Infrared Physics & Technology*, 2018.

[24] M. Pouzet, P. Bonnin, J. Laneurit, and C. Tessier, "A robust real-time image algorithm for moving target detection from unmanned aerial vehicles (uav)," in *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, vol. 1. IEEE, 2014, pp. 266–273.

[25] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[27] C. D. Manning, P. Raghavan, and H. Schütze, "Support vector machines and machine learning on documents," *Introduction to Information Retrieval*, pp. 319–348, 2008.

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[29] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[31] A. Rozantsev, V. Lepetit, and P. Fua, "Flying objects detection from a single moving camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, no. EPFL-CONF-206719, 2015, pp. 4128–4136.

[32] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, pp. 411–426, 2007.

[33] H. Tayara, K. G. Soo, and K. T. Chong, "Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network," *IEEE Access*, vol. 6, pp. 2220–2230, 2018.

[34] K. Liu and G. Mattyus, "Fast multiclass vehicle detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1938–1942, 2015.

[35] F. Tanner, B. Colder, C. Pullen, D. Heagy, M. Eppolito, V. Carlan, C. Oertel, and P. Sallee, "Overhead imagery research data set—an annotated data library & tools to aid in the development of computer vision algorithms," in *Applied Imagery Pattern Recognition Workshop (AIPRW), 2009 IEEE*. IEEE, 2009, pp. 1–8.

[36] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[37] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," Technical report, Tech. Rep., 2017.

[38] W. B. Thompson and T.-C. Pong, "Detecting moving objects," *International journal of computer vision*, vol. 4, no. 1, pp. 39–57, 1990.

[39] S. Aslani and H. Mahdavi-Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 7, no. 9, pp. 1252–1256, 2013.

[40] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[41] M. Yokoyama and T. Poggio, "A contour-based moving object detection and tracking," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on.* IEEE, 2005, pp. 271–276.

[42] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *European conference on computer vision.* Springer, 2010, pp. 282–295.

[43] S. Singh, C. Arora, and C. Jawahar, "Trajectory aligned features for first person action recognition," *Pattern Recognition*, vol. 62, pp. 45–55, 2017.

[44] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.* " O'Reilly Media, Inc.", 2016.

[45] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[46] A. Wedel and D. Cremers, *Stereo scene flow for 3D motion analysis*. Springer Science & Business Media, 2011.

[47] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, pp. 1–10, 2001.

[48] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans on PAMI*, vol. 34, no. 9, pp. 1744–1757, 2012.

[49] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2432–2439.

[50] H. Shen, S. Li, C. Zhu, H. Chang, and J. Zhang, "Moving object detection in aerial video based on spatiotemporal saliency," *Chinese Journal of Aeronautics*, vol. 26, no. 5, pp. 1211–1217, 2013.

[51] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *JACM*, vol. 58, no. 3, p. 11, 2011.

[52] Z. Zhou, X. Li, J. Wright, E. Candes, and Y. Ma, "Stable principal component pursuit," in *ISIT, 2010 IEEE Int Symposium on*. IEEE, 2010, pp. 1518–1522.

[53] S. R. Becker, E. J. Candès, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Math Programming Comp*, vol. 3, no. 3, pp. 165–218, 2011.

[54] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *PAMI, IEEE Trans on*, vol. 35, no. 3, pp. 597–610, 2013.

[55] O. Oreifej, X. Li, and M. Shah, "Simultaneous video stabilization and moving object detection in turbulence," *PAMI, IEEE Trans on*, vol. 35, no. 2, pp. 450–462, 2013.

[56] B. Wohlberg, R. Chartrand, and J. Theiler, "Local principal component pursuit for nonlinear datasets," in *ICASSP, 2012 IEEE Int Conf on*. IEEE, 2012, pp. 3925–3928.

[57] B. Xin, Y. Tian, Y. Wang, and W. Gao, "Background subtraction via generalized fused lasso foreground modeling," *arXiv preprint arXiv:1504.03707*, 2015.

[58] B. Rezaei and S. Ostadabbas, "Moving object detection through robust matrix completion augmented with objectness," *IEEE Journal of Selected Topics in Signal Processing*, 2018.

[59] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European conference on computer vision*. Springer, 2014, pp. 391–405.

[60] G. Tang and A. Nehorai, "Robust principal component analysis based on low-rank and block-sparse matrix decomposition," in *CISS, 2011 45th Annual Conf on*. IEEE, 2011, pp. 1–5.

[61] L. Zhu, Y. Hao, and Y. Song, "$\ell_{1/2}$ norm and spatial continuity regularized low-rank approximation for moving object detection in dynamic background," *IEEE Signal Processing Letters*, vol. 25, no. 1, pp. 15–19, 2018.

[62] B. Shijila, A. J. Tom, and S. N. George, "Simultaneous denoising and moving object detection using low rank approximation," *Future Generation Computer Systems*, vol. 90, pp. 198–210, 2019.

[63] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images," in *CVPR, 2010 IEEE Int Conf on.* IEEE, 2010, pp. 763–770.

[64] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, 2010.

[65] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images," *PAMI, IEEE Trans on*, vol. 34, no. 11, pp. 2233–2246, 2012.

[66] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, pp. 67–68, 1999.

[67] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *JMLR*, vol. 11, pp. 2287–2322, 2010.

[68] R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[69] C. Liu *et al.*, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[70] Z. Gao, L.-F. Cheong, and Y.-X. Wang, "Block-sparse rpca for salient motion detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 10, pp. 1975–1987, 2014.

[71] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Commun, Control, and Comp, 2010 48th Annual Allerton Conf on.* IEEE, 2010, pp. 704–711.

[72] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *CVPR, 2012 IEEE Conf on.* IEEE, 2012, pp. 1568–1575.

[73] H. Mansour and X. Jiang, "A robust online subspace estimation and tracking algorithm," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 4065–4069.

[74] H. Mansour, "A short note on improved roseta," *arXiv preprint arXiv:1710.05961*, 2017.

[75] J. Xu, V. K. Ithapu, L. Mukherjee, J. M. Rehg, and V. Singh, "Gosus: Grassmannian online subspace updates with structured-sparsity," in *ICCV, 2013 IEEE Int Conf on.* IEEE, 2013, pp. 3376–3383.

[76] C. Qiu and N. Vaswani, "Reprocs: A missing link between recursive robust pca and recursive sparse recovery in large but correlated noise," *arXiv preprint arXiv:1106.3286*, 2011.

[77] K. Hoffman and R. Kunze, *Linear Algebra.* Pearson, 1971.

[78] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4284–4297, 2014.

[79] P. Rodriguez and B. Wohlberg, "Fast principal component pursuit via alternating minimization," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013, pp. 69–73.

[80] P. Rodriguez and B. Wohlberg, "Incremental principal component pursuit for video background modeling," *Journal of Mathematical Imaging and Vision*, vol. 55, no. 1, pp. 1–18, 2016.

[81] P. Rodrıguez and B. Wohlberg, "An incremental principal component pursuit algorithm via projections onto the l1 ball," in *XXIV International Congress of Electrical Engineering, Electronics and Computing*, 2017.

[82] G. Chau and P. Rodrıguez, "Panning and jitter invariant incremental principal component pursuit for video background modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1844–1852.

[83] M. Shakeri and H. Zhang, "Corola: a sequential solution to moving object detection using low-rank approximation," *Computer Vision and Image Understanding*, vol. 146, pp. 27–39, 2016.

[84] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 404–412.

[85] I. Markovsky, *Low rank approximation: algorithms, implementation, applications*. Springer Science & Business Media, 2011.

[86] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[87] J. K. Uhlmann, "Dynamic map building and localization: New theoretical foundations," Ph.D. dissertation, University of Oxford Oxford, 1995.

[88] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[89] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[90] D. P. Bertsekas, *Convex optimization theory*. Athena Scientific Belmont, MA, 2009.

[91] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv preprint arXiv:1009.5055*, 2010.

[92] S. C. Eisenstat and H. F. Walker, "Globally convergent inexact newton methods," *SIAM J on Optimization*, vol. 4, no. 2, pp. 393–422, 1994.

[93] R. N. Elias, A. L. Coutinho, and M. A. Martins, "Inexact newton-type methods for non-linear problems arising from the supg/pspg solution of steady incompressible navier-stokes equations," *J of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 26, no. 3, pp. 330–339, 2004.

[94] N. Krejic and J. M. Martínez, "A globally convergent inexact-newton method for solving reducible nonlinear systems of equations," *Optimization methods and Software*, vol. 13, no. 1, pp. 11–34, 2000.

[95] R. Collins, X. Zhou, and S. K. Teh, "An open source tracking testbed and evaluation web site," in *IEEE Intern Workshop on Performance Evaluation of Tracking and Surveillance*, 2005, pp. 17–24.

[96] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *CVIU*, vol. 117, no. 6, pp. 633–659, 2013.

[97] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR, 2011 IEEE Conf on*. IEEE, 2011, pp. 3153–3160.

[98] K. L. Veon, M. H. Mahoor, and R. M. Voyles, "Video stabilization using sift-me features and fuzzy clustering," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2377–2382.

[99] T. Bouwmans and E. H. Zahzah, "Robust pca via principal component pursuit: a review for a comparative evaluation in video surveillance," *CVIU*, vol. 122, pp. 22–34, 2014.

[100] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *ICCV 2005. Tenth IEEE Int Conf on*, vol. 2. IEEE, 2005, pp. 1508–1515.

[101] J. Lewis, "Fast normalized cross-correlation," in *Vision interface*, vol. 10, no. 1, 1995, pp. 120–123.

[102] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact newton methods," *SIAM J on Numer anal*, vol. 19, no. 2, pp. 400–408, 1982.

[103] S. C. Eisenstat and H. F. Walker, "Choosing the forcing terms in an inexact newton method," *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 16–32, 1996.

[104] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[105] M. Ghanbari, "The cross-search algorithm for motion estimation (image coding)," *IEEE Transactions on Communications*, vol. 38, no. 7, pp. 950–953, 1990.

[106] M. Shakeri and H. Zhang, "Corola: A sequential solution to moving object detection using low-rank approximation," *arXiv preprint arXiv:1505.03566*, 2015.