AFRL-AFOSR-VA-TR-2022-0013

Real Time Computer Vision

Kallemov, Bakytzhan
Novateur Research Solutions LLC
20452 Scioto Ter
Ashburn, VA, 20147-7008
US

10/27/2021
Final Technical Report

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
27-10-2021	Final	15 Sep 2018 - 14 Sep 2021

4. TITLE AND SUBTITLE
Real Time Computer Vision

5a. CONTRACT NUMBER
FA9550-18-C-0050

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Bakytzhan Kallemov

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Novateur Research Solutions LLC
20452 Scioto Ter
Ashburn, VA 20147-7008
US

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTB1

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2022-0013

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The SBIR Phase II effort developed computer vision and machine learning algorithms that exploits algebraic constraints on the background appearance as well as motion to differentiate foreground regions from background. By incorporating sensor motion with the appearance model, the developed approach is able to handle both static and moving cameras. The Phase II effort includes; development of background separation algorithms and enabling technologies, development of software, and quantitative and qualitative evaluation of the technologies using real-world data.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			ARJE NACHMAN
U	U	U	UU	50	**19b. TELEPHONE NUMBER** *(Include area code)* 426-8427

Standard Form 298 (Rev.8/98)
Prescribed by ANSI Std. Z39.18

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
14-10-21	Final	15 September 2018 – 14 September 2021

4. TITLE AND SUBTITLE
Real Time Computer Vision

5a. CONTRACT NUMBER
FA9550-18-C-0050

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Khurram Hassan-Shafique

5d. PROJECT NUMBER

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Novateur Research Solutions LLC
20110 Ashbrook Place, Suite 275
Ashburn, VA 20147

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
875 North Randolph Street
Arlington, VA 22203-1954

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFOSR

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution authorized to U.S. Government Agencies only; contains proprietary Information (SBIR Data Rights); April 2017. Other requests for this document shall be referred to Air Force Office of Scientific Research.

13. SUPPLEMENTARY NOTES
This is a Small Business Innovation Research (SBIR) Phase II Report. Report contains color.

14. ABSTRACT
This report was developed under a SBIR contract for topic AF151-001. The Phase II effort developed computer vision and machine learning algorithms that exploits algebraic constraints on the background appearance as well as motion to differentiate foreground regions from background. By incorporating sensor motion with the appearance model, the developed approach is able to handle both static and moving cameras. The Phase II effort includes; development of background separation algorithms and enabling technologies, development of software, and quantitative and qualitative evaluation of the technologies using real-world data.

15. SUBJECT TERMS
SBIR Report, background separation, object detection, computer vision, real-time sensor exploitation, automated visual surveillance

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Arje Nachman
Unclassified	Unclassified	Unclassified	SAR	48	19b. TELEPHONE NUMBER *(Include Area Code)* (703) 696-8427

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Real Time Computer Vision (Phase II)

Final Technical Report

REPORTING PERIOD:	September 15, 2018 – September 14, 2021
Project Sponsored by:	Department of the Air Force
Contract Number:	FA9550-18-C-0050
Principal Investigator:	Mr. Maoxu Li
Contractor:	Novateur Research Solutions LLC 20110 Ashbrook Place, Suite 275 Ashburn, VA 20147 703-468-1200

Novateur Research Solutions LLC,
20110 Ashbrook Place, Suite 275, Ashburn, VA 20147
www.novateur.ai

Table of Contents

Table of Figures

Glossary

BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Network
CSI	Camera Serial Interface
DMD	Dynamic Mode Decomposition
FAST	Features from accelerated segment test
FgSegV2	Foreground Segmentation Network Version 2
FMV	Full Motion Video
FPS	Frames Per Second
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRASP	General Robotics, Automation, Sensing and Perception
GRASTA	Grassmannian Robust Adaptive Subspace Tracking Algorithm
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ISR	Intelligence, Surveillance, and Reconnaissance
ISVD	Incremental Singular Value Decomposition
KDE	Kernel Density Estimation
LSF	Least Square Fit
LU	Lower-Upper
MRA	Multi-resolution Analysis
MRDMD	Multiresolution DMD
NVENC	NVIDIA Encoder
ORB	Oriented Fast and Rotated BRIEF
PC	Personal Computer
PCA	Principal Component Analysis
PIMC	Polar Incremental Matrix Completion
PRMF	Probablistic Robust Matrix Factorization
PTZ	Pan-Tilt-Zoom
PWC-Net	Pyramid, Warping, and Cost Volume Network
RAM	Random Access Memory
RANSAC	Random Sample Consensus
ResNet	Residual Network

ROS	Robotic Operating System
RPCA	Robust Principal Component Analysis
RTCV	Real-Time Computer Vision
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SDK	Software Development Toolkit
SVD	Singular Value Decomposition
SWAP	Size, Weight, and Power
SURF	Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicle
UCF	University of Central Florida
VIC	Video Image Compositor

1. Summary

Background separation of foreground objects from background is the first step in many video processing pipeline. While existing approaches for background separation adequately perform to solve many of the technical challenges, such as illumination changes, dynamic backgrounds (e.g., fluttering leaves, waving flags, etc.), camera jitter, etc.), they have limitations in handling mobile sensors and also have high computational complexity. This is a major limitation for future intelligent systems due to recent prevalence of mobile sensors and small platforms requiring onboard processing with size, weight, and power (SWaP) constraints.

This Small Business Innovation Research (SBIR) effort made several innovations to handle these challenges and to address the limitations of the state-of-the-art: These include i) development of innovative mathematical models and advanced algorithms based on that effectively handle challenging cases for efficient background subtraction on small platforms; and ii) development of mathematical models of viewing geometry of moving sensor and image formation and exploiting the low-rank constraints of rigid body motion in 3D world for efficient detection of independently moving objects. The Phase II effort developed computer vision and machine learning algorithms that exploits low-rank constraints on the background appearance as well as motion to differentiate foreground regions from background. The low-rank constraints on the background appearance are based on both the theoretical and empirical results that show that the vectorized images corresponding to a given object under different transformations (e.g., illumination variations) approximately lie on a low dimensional subspace. Moreover, the low-rank constraints on motion exploit viewing geometry of freely moving sensors to differentiate foreground and background motion. By incorporating sensor motion with the appearance model, the developed approach is able to handle both static and moving cameras. The background separation algorithm was also ported to a mobile device to provide real-time performance and evaluated on both standard and in-house datasets. The technologies developed during the project provide a natural framework and foundation for robust and efficient automated computer vision on low-cost mobile devices.

2. Introduction

The prime objective of the proposed effort is to develop technologies that enable real-time background/foreground separation in static and freely moving video sensors using mobile processors.

Figure 1: *The proposed background separation technologies will enable exploitation of video sensors on mobile platforms and have wide applications that include: aerial video exploitation, augmented reality and gaming, edge processing of high resolution videos, wearable sensors, and autonomous vehicles.*

Video sensors have become a ubiquitous source of information in all domains of life (Figure 1). From defense to commercial to consumer sector, they are increasingly being used to enable a wide range of applications. For example, fixed and pan-tilt-zoom (PTZ) cameras are commonly used to provide visual surveillance, critical infrastructure security, business intelligence, and building automation. Similarly, cameras mounted on UAV platforms are easily one of the most critical sources of intelligence, reconnaissance, and surveillance (ISR) data and situational awareness. In addition, consumers around the world are generating a large amount of video data from handheld devices (e.g., smart phones) and wearable devices (e.g, Google Glass, GoPro, etc.).

The large amount of data generated by these sensors has pushed the demand for automated techniques that can extract useful information from rich video data. For example, automated visual surveillance systems and aerial video exploitation tools are widely used to detect and track targets of interest and recognize activities from video. However, most of the existing visual analysis systems assume availability of large processing infrastructure (powerful PCs, servers, GPUs, etc.) and therefore cannot be easily applied to many of the new mobile sensing devices, or small platforms with strict size, weight and power (SWaP) constraints. Therefore, there is a need to develop efficient computer vision technologies that enable real-time exploitation of videos using low-power low-cost mobile computing platforms that can be easily integrated with the sensing device.

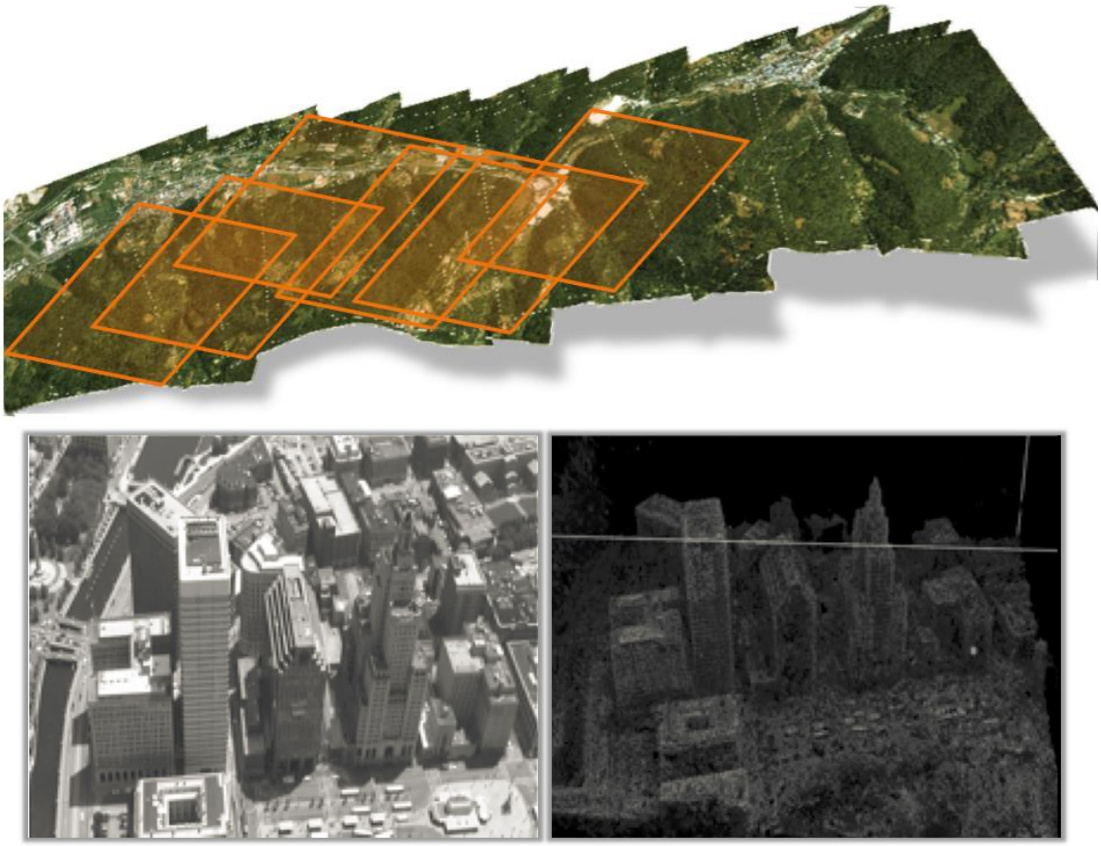

Figure 2: *Mosaic (top) and voxel-based (bottom) representations used by state-of-the-art methods for background modeling are computationally very expensive and infeasible for mobile platforms.*

2.1 Limitations of the State-of-the-Art Background Separation Technologies:

The first step in many video processing pipelines is separation of foreground objects from background. This is done through background separation algorithms that attempt to identify the most relevant parts of the video stream by adaptively learning and modeling the characteristics of background and finding pixels that do not conform to the learned model. As one of the most critical components of automated visual surveillance systems, background separation problem has been extensively studied in computer vision literature. Many approaches have been presented to solve its technical challenges, such as illumination changes, dynamic backgrounds (e.g., fluttering leaves, waving flags, water fountains, etc.), camera jitter, shadows, and moving camera, etc. While existing methods are quite adept at handling many of these challenges, they also have high computational complexity (in terms of number of pixels) and are not suitable for mobile platforms.

On the other hand, a large portion of video data generated these days is captured by mobile sensors, e.g., handheld smart phones, wearable devices (GoPro, Google Glass, etc.), and sensors mounted on small UAVs. However, most of the existing work in background separation has focused on the videos from static and PTZ cameras used in video surveillance. This holds true even for some of the most recent algorithms that use advanced online optimization and subspace estimation

techniques [GK15, HBS12] for fast background separation. Therefore, these algorithms cannot be directly applied to videos from moving sensors.

The methods that do attempt to solve the problem for mobile sensors also borrow the basic methodology from algorithms for static cameras. These methods first create new representations (Figure 2) that cancel the effect of platform motion, e.g., background mosaics (generated by stitching imagery from subsequent frames) or explicit 3D models (created using structure-from-motion and stereopsis techniques). The background separation techniques borrowed from static camera domain are then applied to these new representations. In addition to the complexity of original separation techniques, this methodology introduces even more computationally complex elements to the processing pipeline. For example, image-stitching algorithms required to do mosaic generation involves very expensive image warping (transformation) components. Similarly, structure-from-motion and stereopsis techniques and the resulting 3D voxel-representations cannot be created in real-time without heavy parallel processing using GPUs.

2.2 Summary of Innovation:

To handle the above challenges, Novateur Research Solutions and General Robotics, Automation, Sensing and Perception (GRASP) Laboratory at University of Pennsylvania, herein after referred to as the Novateur Team made several mathematical innovations during the SBIR effort. These included, development of innovative mathematical models as well as advanced computer vision and machine learning techniques that can effectively separate background from foreground from both moving and stationary sensors from a wide variety of scenarios.

In particular, during the Phase II effort, the team developed algebraic models that exploit low-rank constraints on the background appearance as well as motion to differentiate foreground regions from background. The low-rank constraints on the background appearance are based on both the theoretical and empirical results that show that the vectorized images corresponding to a given object under different transformations (e.g., illumination variations) approximately lie on a low dimensional subspace. Moreover, the low-rank constraints on motion exploit viewing geometry of freely moving sensors to differentiate foreground and background motion. The team then developed novel algorithms both based on algebraic optimization (matrix factorization) as well as machine learning (neural networks) that exploit these constraints for background subtraction from moving platforms.

The background separation technologies developed during the Phase I are

- do not require creation of expensive scene representations;
- do not make assumptions about the scene or platform motion; and
- agnostic to specific computing architectures or instruction sets.

Therefore, they enable efficient background separation in high-resolution imagery from both static and moving sensors using a wide variety of low-cost, low-power, and light-weight processing units.

2.3 Phase II Accomplishments:

The key objectives that were accomplished during Phase II research and development effort include:

A Novel Algebraic Framework for Efficient Online Extraction of Multiple Moving Targets in Moving Sensor Videos – During the Phase I effort, the Novateur Team approached the problem of discovering and segmenting independently moving objects by casting it as a low-ranking approximation problem. More specifically, the techniques that we developed involve detecting and tracking features between frames and then aggregating those tracking results in incomplete matrices where the missing entries reflect the fact that features will become occluded over time as the camera moves through space. We then showed that we could exploit the assumption that most of the scene was moving rigidly and the structure of the affine projection model which mandate that most of these tracks must lie on a three-dimensional manifold in a high dimensional space. The next key idea in our approach was to exploit online incomplete matrix factorization techniques to quickly and efficiently extract this three-dimensional subspace structure from the measurements. Once this has been done, independently moving obstacles can be recovered by identifying outliers to the low-rank model then grouping these features based on proximity in the images. The Phase II effort further built upon the Phase I model by incorporating subspace tracking [HBS12] and Dynamic Mode Decomposition [GK15, KFB15] and using the low-rank property of the background model to extract foreground pixels in real-time. In addition, the Phase II effort developed novel machine learning model that exploit the low-dimensional constraint to separate the static 3D scene geometry from independently moving objects.

Improved Moving Target Detection on Mobile Computing Platform using Background Separation Technology as a Focus-of-Attention Mechanism – The Phase II effort developed a novel computer vision algorithm that is capable of detecting moving target detection on mobile platforms, such as UAVs. The algorithm combines optical flow, deep neural networks, as well as low-rank constraints and provides focus-of-attention mechanisms for further processing tasks.

Testing, and Evaluation – The Novateur Team also demonstrated capabilities of the proposed background separation technologies using standard benchmark datasets as well as in-house datasets consisting of real-world scenarios. The Team also performed quantitative and qualitative evaluation of the proposed technologies, which included performance characterization and trade-off analyses on real-world data.

Implementation of an End-to-End Software to Enable Robust Detection of Moving Targets in Videos – The Phase II implemented an end-to-end software based on ROS architecture to enable robust detection of targets in video from static and moving sensors using mobile computing platforms.

3. Methods, Assumptions, Procedures, and Results

3.1 Project Foundation and Planning:

In the beginning of the project, the team performed initial task planning. This involved internal kickoff meeting to agree upon the task assignments and work plan. It also involved initial data and existing software gathering to support development and evaluation of the proposed technologies. The team also analyzed the existing datasets and developed foundation algorithms and tools to support algorithm development and testing.

3.2 Algebraic Framework for Efficient Online Extraction of Multiple Moving Targets in Moving Sensor Videos

3.2.1 Background:

The use of low-dimensional subspace as background model enables use of online matrix completion apparatus to enable robust and efficient background separation. Low-rank matrix completion is the problem of recovering a low-rank matrix from an incomplete sample of the entries. During the Phase I of this effort, we used a new algorithm called Polar Incremental Matrix Completion (PIMC) to develop a background subtraction approach using online subspace tracking.

Online Subspace Tracking Problem: Given a sequence of d-dimensional subspaces, $\mathcal{S}_t \subset \mathbb{R}^n, d < n$, and a sequence of vectors $v_t \in \mathcal{S}_t$. The objective of a subspace tracking algorithm with subsampled vectors is to estimate $\mathcal{S}_t$ given v_{Ω_t}- an incomplete version of v_t, sampled only on the indices $\Omega_t \subset \{1,2,\dots,n\}$. Let the columns of an $n \times d$ matrix U_t be orthonormal and span $\mathcal{S}_t$. Tracking the evolving subspace $\mathcal{S}_t$ is equivalent to estimating U_t at each time step.

Modeling of Background as a Low-Rank Subspace: The first step in background separation is modeling of background and foreground attributes. Given these models, the background separation for each incoming set of observations amounts to finding the likelihood ratio of each observation with respect to background and foreground models. Therefore, the choice of models, the attributes being modeled, and the mathematical framework used is important for both the quality and the efficiency of the background separation process. To be applicable in a wide variety of real-world applications, background models should not use assumptions that are specific to a domain or application and must have some desirable properties, for example, they should be robust to illumination changes, background clutter, and dynamic backgrounds (e.g., fluttering leaves, waving flags, water waves, etc.). Moreover, since the background attributes keep changing over

time, the models also need to be adaptive and should evolve over time by actively learning from sensor observations.

A common characteristic used for modeling background appearance is that the attributes (e.g., color, gradient, texture etc.) of the image locations belonging to objects of interest change more rapidly than those belonging to background scene. Existing background separation methods model this characteristic by maintaining probability density models (e.g., mixture of Gaussians, parzen windows, etc) for each image location. For each incoming observation (video frame), the distance between the observation and the probability density is used as a likelihood of the observation being part of the background. This method is computationally inefficient. For an image of dimensions N × M pixels, this method requires maintenance of NM probability distributions and independent likelihood computations. It treats each density independently and therefore does not exploit strong correlations between different observations.

It has been shown both theoretically and empirically [BJ03, ZM+08, HBS12] that the vectorised images corresponding to a given object or scene under different conditions (e.g., illumination) approximately lie on a low-dimensional subspace. Using this observation, we can make the background models more powerful (and efficient) by keeping a d-dimensional subspace or affine set with $d > 0$. Suppose we have T frames and $V = [v_1, v_2, \dots, v_T]$, where each length-$NM$ vector v_i corresponds to the $N \times M$ pixels of the ith image in the sequence. A d-dimensional subspace model is then a factorization $V \approx LR^T$, where L is an $NM \times d$ basis matrix, and R is a $T \times d$ coefficient matrix. At each time step t, we assume that a vector or video frame v_t is generated by the following model: $v_t = U_t w_t + s_t + \zeta_t$, where U_t is the background subspace, w_t is the $d \times 1$ weight vector, ζ_t is the $n \times 1$ zero-mean Gaussian white noise vector with small variance, and s_t is the $n \times 1$ sparse outlier vector that models foreground pixels in the separation problem. The problem of background separation can then be modeled as that of minimizing the objective function $\arg\min_{w_t}(\|U_t w_t - v_t\|)$ subject to rank constraint on background matrix U_t (where $\|\cdot\|_1$ is the l^1 norm). Let U_{Ω_t} denote the submatrix of U_t consisting of the rows indexed by Ω_t; also for a vector $v_t \in \mathbb{R}^n$, let v_{Ω_t} denote a vector in $\mathbb{R}^{|\Omega_t|}$ whose entries are those of v_t indexed by Ω_t. For each time step t, we observe v_{Ω_t}, a subsampled vector (by sampling pixels from each incoming video frame). The subspace error from the subspace spanned by the columns of U_t to the observed vector v_{Ω_t} is given by $\min_{w_t}\left(\left\|U_{\Omega_t} w_t - v_{\Omega_t}\right\|\right)$. We estimate the subspace U_t using these sampled vectors and track it over time. The sparse outlier vector at each step provides the estimate of the foreground.

3.2.2 Background Separation using Dynamic Mode Decomposition:

During Phase II research work we applied recently developed methods, such as subspace tracking [HBS12] and Dynamic Mode Decomposition (DMD) [GK15, KFB15] and using the low-rank property of the background model to extract foreground pixels at frame-rates that are 2-3 orders

of magnitude faster than conventional methods. The algorithm was implemented in C++ with OpenMP multithreaded support. The Figure 3 demonstrates the successful application of a dynamic mode decomposition method to eliminate parallax effects for the UAV video.

Figure 3: *Illustration of DMD method for feature based and direct method for UAV video. Left: original input images from video sequence; Center: feature based DMD separation of background/foreground (red blobs indicates foreground features consistent across frame window, blue blobs are only for the current frame); Right: Foreground mask for the direct method using DMD.*

Dynamic Mode Decomposition:

Dynamic Mode Decomposition (DMD) method has been originally introduced in the computational fluid dynamics field, DMD has emerged as a powerful tool for analyzing the dynamics of nonlinear systems, including neuroscience and financial trading. Given data collected in regularly space time intervals, the DMD method will approximate the low-dimensional modes of the linear, time-independent Koopman operator in order to estimate the potentially nonlinear dynamics of the system. Recently, it has been proved to be a very powerful tool in video streams for tracking objects and to separate background from dynamic foreground [KFB15, BYZ18]. In recent work DMD is used to successfully for shot boundary detections [BYZ+18]. One of the key advantages that DMD method is that it is not only a data-driven method that provides a decomposition into spatial-temporal modes (similar to the principal component analysis PCA), but also correlates data to unique temporal Fourier modes. Thus, a matrix decomposition is done both in time and space allowing the method to be well suited to motion detection, which separated video data into low-rank(background) and sparse(foreground) components.

Multi-resolution Dynamic Mode Decomposition (MRDMD):

Adding into the consideration a windowed Fourier transform, one can systematically remove temporal or spatial features by a process of recursive refinement of sampling from the data of interest, so to perform multi-resolution analyses (MRA). Integrating this concept to DMD we can

formulate a Multi-resolution DMD (MRDMD) [KFB15]. To apply MRDMD method we assume that we can reconstruct a dynamical system that is comprised of multi-scale temporal and/or spatial features. This allows us easily separable multi-scale spatio-temporal features in MRDMD decomposition and provide a powerful tool to analyze of video feeds in a real-time architecture. Consider a dynamical system with two sets of data $\boldsymbol{X} = [\boldsymbol{x_1}\ \boldsymbol{x_2}\ \ldots \boldsymbol{x_M}]$, $\boldsymbol{X'} = [\boldsymbol{x'}_1\ \boldsymbol{x'}_2\ \ldots \boldsymbol{x'}_M] \in \mathbb{R}^{\boldsymbol{N}\times\boldsymbol{M}}$, $\boldsymbol{x_k}$ and $\boldsymbol{x'}_k$ are vector of N data points of the frame k collected at initial and after some time τ accordingly. Define a matrix $\boldsymbol{A} = \boldsymbol{X'X}^{\dagger}$, where † denotes Moore-Penrose pseudoinverse. The DMD procedure constructs the proxy, approximate linear evolution

$$\frac{d\tilde{\boldsymbol{x}}}{dt} = \boldsymbol{A}\tilde{\boldsymbol{x}}, \quad \tilde{\boldsymbol{x}} = \sum_{\boldsymbol{k}}^{\boldsymbol{M}} b_k\, \psi_k \exp\,(\omega_k t)$$

where ψ_k , ω_k are eigenfunctions and eigenvalues of the matrix $\boldsymbol{A}$. In practice the dimension N can be large to analyze the matrix $\boldsymbol{A}$ directly. DMD method consider a rank-reduced representation of a projected matrix $\tilde{\boldsymbol{A}}$.

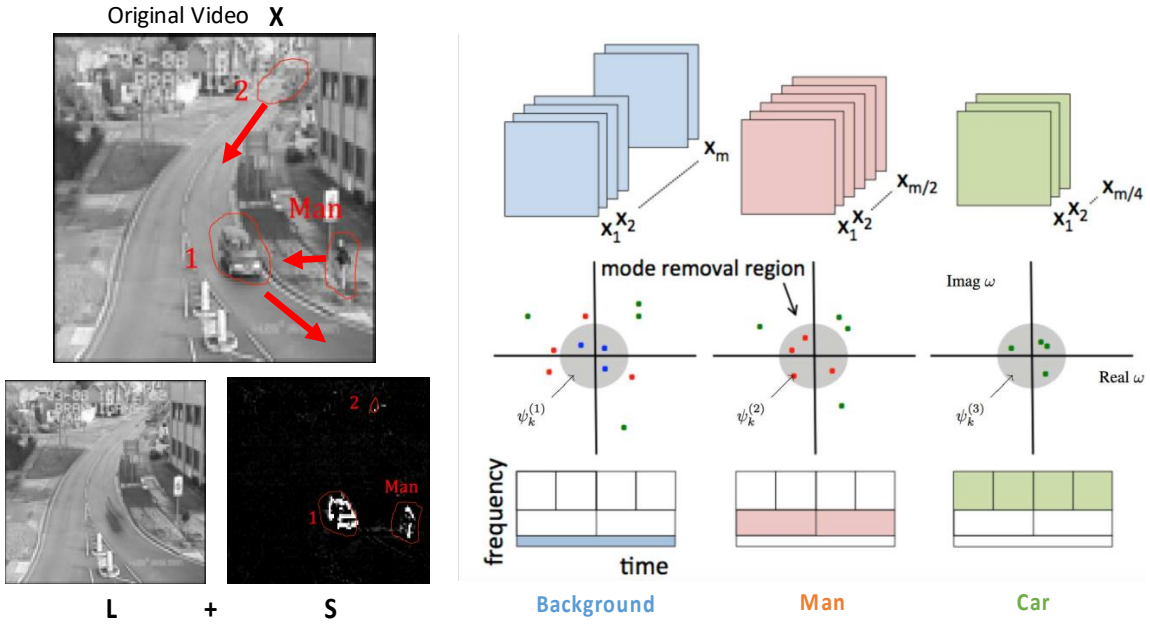

Figure 4: *Representation of the multi-resolution dynamic mode decomposition on an example video that includes three different time scale features (annotated), a background, a pedestrian (slow) and a car (fast) (from [KFB15]).*

The DMD algorithm proceeds as follows:

1) Decompose matrix $\boldsymbol{X}$ via reduced SVD with rank K $\boldsymbol{X} = \boldsymbol{U\Sigma V}^{*}$;

2) Compute $\boldsymbol{A} = \widetilde{\boldsymbol{U}^{*}\boldsymbol{X}'\boldsymbol{V}}\boldsymbol{\Sigma}^{-1}$, the $K \times K$ projection of the full matrix;

3) Find eigenvectors $\boldsymbol{W}$ and eigenvalues $[\lambda_1 \ \lambda_2 \ldots \lambda_K]$ of the matrix $\widetilde{\boldsymbol{A}}$; 4) Find eigenfunctions $\boldsymbol{\Psi} = [\boldsymbol{\psi_1} \ \boldsymbol{\psi_2} \ \ldots \boldsymbol{\psi_K}] = \boldsymbol{X'V\Sigma^{-1}W}$ and eigenvalues $\omega_k = \ln(\lambda_k)/\Delta t$, with Δt is a time interval between frames. Then approximate solution at time t is given $\tilde{\boldsymbol{x}}(\mathrm{t}) = \sum_{\boldsymbol{k}}^{\boldsymbol{K}} b_k \, \psi_k \exp(\omega_k t)$, where $b_k = \boldsymbol{\Psi}^{\dagger}\boldsymbol{x_1}$.

To separate different time-scale features, we decompose the given data into low-rank background $\boldsymbol{L_i}$ and sparse $\boldsymbol{S_i}$, so $\boldsymbol{X_i} = \boldsymbol{L_i} + \boldsymbol{S_i}$. For each level i, the first m_i modes are removed.

$$\boldsymbol{x_i^{mrDMD}} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{m_i}} b_k \, \psi_k \exp(\omega_k t) + \sum_{\boldsymbol{k=m_i+1}}^{\boldsymbol{M}} b_k \, \psi_k \exp(\omega_k t)$$

(slow modes) (fast modes)

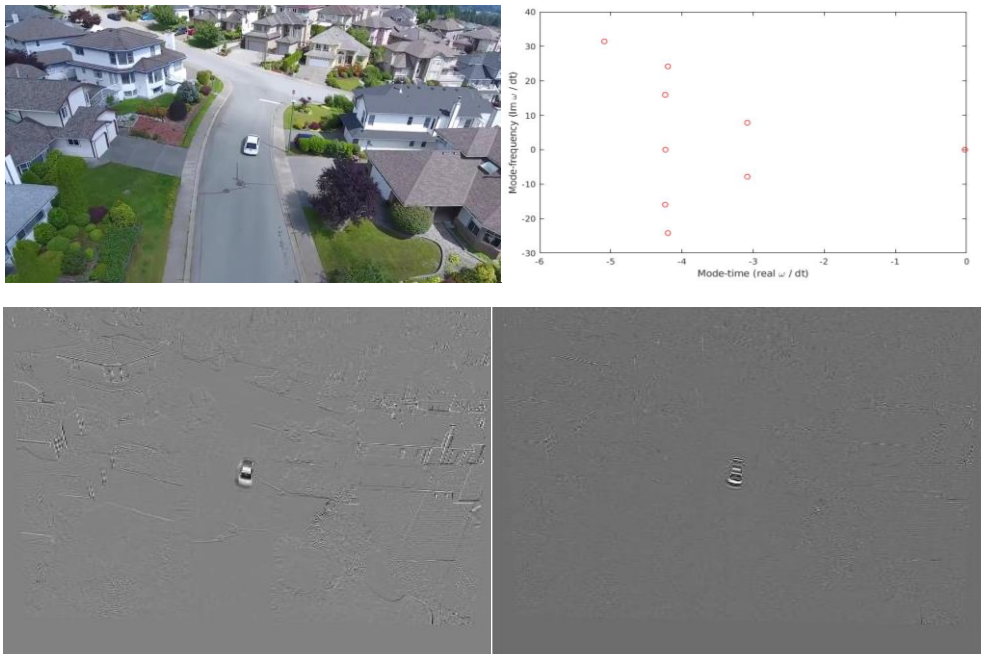

Figure 5: *DMD decomposition with planar rectification for UAV video.Top-left: Input image; Top-right: Fourier modes for DMD; Bottom-left: reconstruction for the foreground with* $\boldsymbol{\epsilon = 1}$ *with remaining parallax effects; Bottom-right: reconstruction for the foreground with* $\boldsymbol{\epsilon = 25}$.

By setting a threshold ϵ for background separation, so that there are only included modes with $real(\omega_k/\Delta \mathrm{t}) < \epsilon$, where Δt is the time between frames. Since we expect that background can be approximated by low rank representation, we can effectively separate the foreground by setting appropriate threshold (Figure 5).

Modes that reside near the origin represents dynamics that are unchanging or changing slowly and can be interpreted as background or low-rank components. For the MrDMD method we can formulate a foreground separation as a difference $\boldsymbol{X_j^{Sparse\ DMD}} = \boldsymbol{X_j} - \left|\boldsymbol{X_j^{Low\ rank\ DMD}}\right|$. However,

we can extend our multi-resolution analysis by applying another DMD for the sparse fast modes dynamics only. For instance, if m_1 modes were used for the background representation, then define $\mathbf{X_{M/2}} = \sum_{\boldsymbol{k=m_1+1}}^{\boldsymbol{M}} b_k\, \psi_k \exp(\omega_k t)$. Applying DMD on $\mathbf{X_{M/2}}$ again we can separate it to $\mathbf{X_{M/2}} = \mathbf{X_{M/2}}^{(1)} + \mathbf{X_{M/2}}^{(2)}$. This procedure can be applied recursively. Then

$$\widetilde{\boldsymbol{x(t)}} = \sum_{\boldsymbol{k}}^{\boldsymbol{m_1}} b_k\, \psi_k \exp(\omega_k t) + \sum_{\boldsymbol{k}}^{\boldsymbol{m_2}} b_k{}^{(1)}\, \psi_k{}^{(1)} \exp(\omega_k t) + \sum_{\boldsymbol{k}}^{\boldsymbol{m_3}} b_k{}^{(2)}\, \psi_k{}^{(2)} \exp(\omega_k t) + \cdots.$$

It allows us to represent modes and their positions in the decomposition structure [KFB15]. The advantage of this approach that it can remove a potential dominance of a single set of modes and provide a different spatio-temporal DMD analysis to obtain a real multi-resolution technique. Therefore, we can extract spatio-temporal structure within the whole frame sequence for shorter time windows, and reconstruct different time-scale features.

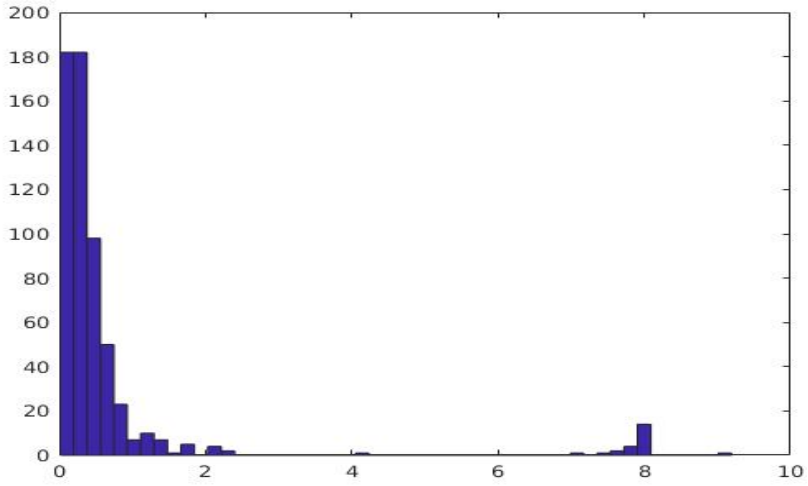

Figure 6: *Error distribution for the background modeling for the UAV video for feature based approach. The outlier points correspond to the moving car.*

Feature Tracking:

The direct method for DMD approach with planar rectification requires a computationally expensive warping procedure. To represent background in moving sensor videos, the scene is modeled by a sparse set of feature points. The feature points can be either randomly sampled on the image or obtained by using feature detection methods such as SURF, BRIEF, FAST and ORB etc. that are designed for efficient performance on low power processors. The feature points are independently tracked over time and their (short-term and possibly noisy) trajectories are maintained with the model (Figure 6).

New points are also added in the model as the sensor moves and observes new scene regions. This sparse representation has several advantages over traditional mosaic and voxel-based representations. As it is based on sparse features that are independently tracked, it is space-efficient and easy to maintain. It does not require expensive image warping or structure estimation and is therefore quite efficient. Our experiments show a significant speed up in the processing compared

to the direct method based on pixels intensities of a warped frame (40-50 fps for 1280x720 video with FAST features detector on regular 4-cores laptop computer).

Figure 7: *Examples of feature based background/foreground separation for UAV video (left) and dash camera video (right).*

3.2.3 Development of Neural Networks that Learn Low-Dimensional Non-Linear Manifolds to Estimate Independently Moving Objects from Moving Platforms

The task of estimating independently moving objects from monocular video imagery is a challenging one since it involves distinguishing between the motion in the image due to the ego motion of the observer that induced by the motion of the objects. In the previous subsection we showed approaches to this problem that tackled this problem by modeling the motion of the background in terms of a low dimensional affine manifold which could be reconstructed using robust regression techniques. Once the background motion had been extracted, foreground objects could be discovered by identifying outliers that did not jibe with the recovered foreground model.

This approach works well in situations where the motion model can be adequately captured by a low dimensional linear subspace. This is a reasonable model for aerial imagery acquired from high flying drones but is less accurate as the distance between the scene and the observer is smaller. There the non-linearities induced by the perspective motion model become more apparent.

In order to address this problem one of the thrusts of our research effort has been to ask whether it is possible to train a neural network to propose a low-dimensional non-linear manifold that captures the depth structure of the scene and then use that to model the foreground motion in the scene. This work builds on and generalizes the body of research that seeks to estimate the depths in a single color image.

Recent learning based monocular depth estimation approaches tried to solve this problem by leveraging the expressive power of deep convolutional neural networks (CNN) with strong regularizations. Despite the fast progress in the field, predicting depth from a single monocular

image remains a fundamentally hard problem. On the other hand, with the advance of active sensing technology, time-of-flight sensors and Lidars are becoming more prevalent. The depth completion problem is then to estimate a dense depth image from a set of sparse depth measurements, often guided by a color image [US+17].

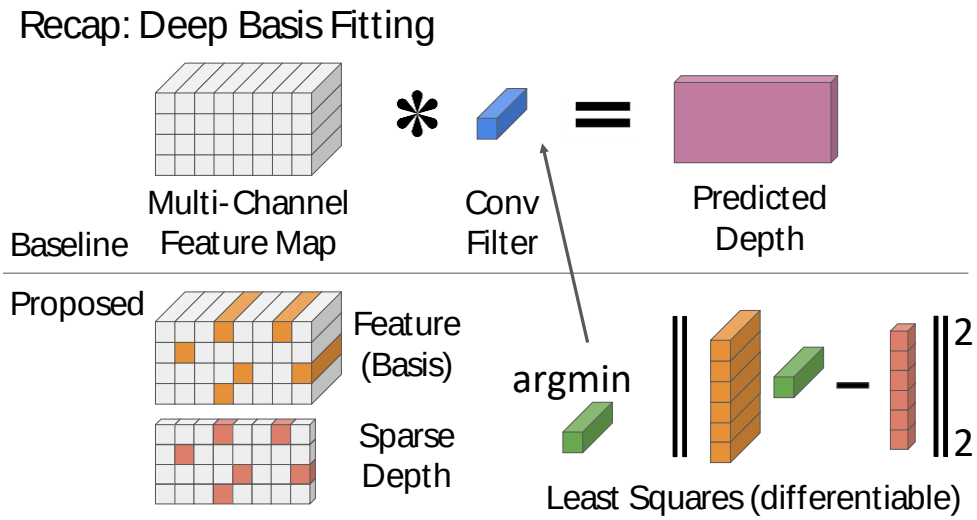

Figure 8: *Overview of deep basis fitting approach where the final convolutional stage is replaced by a differentiable least squares fitting procedure.*

Most depth completion framework opt for feeding the color image together with the sparse depth into a neural network directly and output one single depth image [MCK18]. As this is the most obvious approach, it suffers from the fact that traditional convolutional filters are not designed for sparsity invariance [US+17], and thus have trouble coping with different levels and patterns of sparsity. Instead of giving the network a demanding task which is to directly produce depth, we ask the network to predict a set of basis depth maps from the image. The final depth is then reconstructed from the predicted basis, with the help of the sparse depth input. This approach has the following advantages:

1) we no longer need to worry about input sparsity, since the sparse input are not directly fed into the network,

2) we can handle huge changes in input sparsity without significant performance degradation, as each sparse depth measurement puts one constraint on the basis depth.

Note that an important byproduct of our approach is a network that predicts a depth basis which serves as a low-dimensional structural model which could also be used for motion estimation. To do this we would optimize over the motion parameters between frames and the weights applied to the depth bases to minimize the photometric error observed when one frame is mapped into another.

This approach has the following advantages:

1) we no longer need to worry about input sparsity, since the sparse input are not directly fed into the network,

2) we can handle huge changes in input sparsity without significant performance degradation, as each sparse depth measurement puts one constraint on the basis depth.

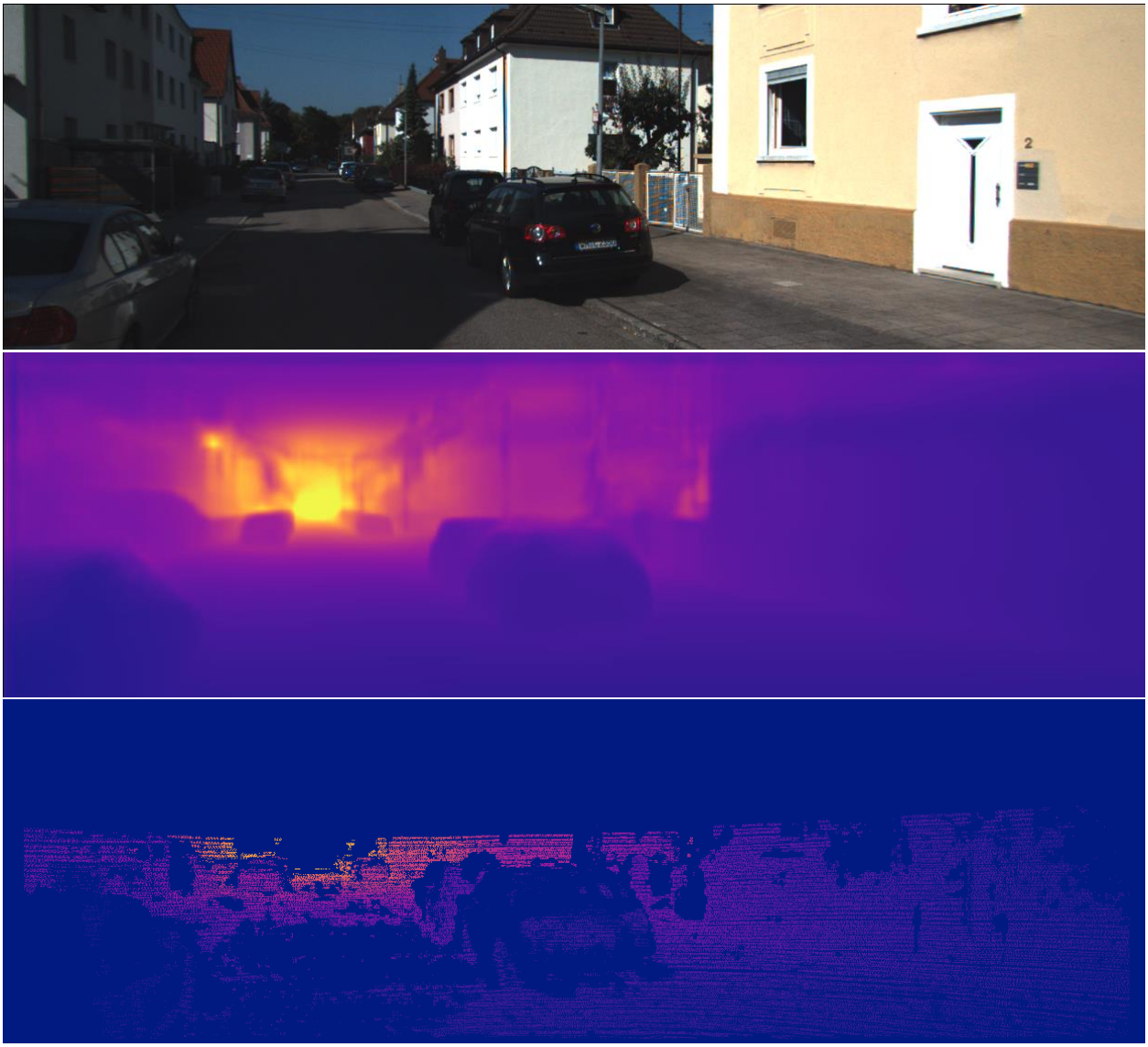

Figure 9: *Top to Bottom: image, depth prediction, groundtruth*

Deep Basis Fitting

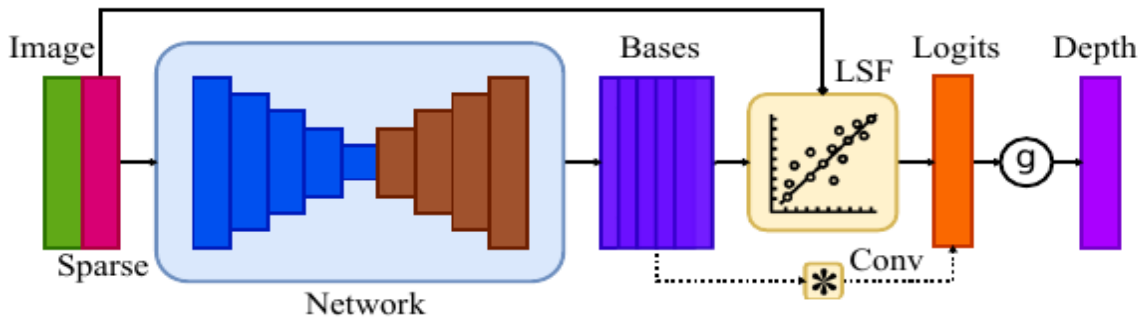

Figure 10: *Deep Basis Fitting architecture. Solid lines indicate the data flow of our module, while dotted lines indicate that of the baseline method, which is simply a convolutional layer. LSF module replaces the convolutional layer with no change to the rest of the network.*

Given an image X and a sparse depth map S, we wish to predict a dense depth image D from a depth estimation function f that minimizes some loss function L with respect to the ground truth depth D. Typically, X is a color image, S the sparse depth map where invalid pixels are encoded by 0, and f a fully convolutional neural network whose parameters are denoted by θ. When ground-truth depth D is available, the learning problem is to determine θ according to

$$\hat{\theta} = \underset{\theta}{argmin} L(f(X, S; \theta), D)$$

Existing depth prediction networks usually employ a final convolutional layer to convert an M-channel set of basis features, B, to a single-channel result, L, which is sometimes referred to as the logits layer. The inputs to this final layer are allowed to range freely between $-\infty$ and $+\infty$ while the logit outputs are mapped to positive depth values by a nonlinear activation function, g. The final convolution filter that maps the basis features, B, onto the logits, L. L is, therefore, an affine combination of channels in B and the predicted depth at pixel i is

$$D'[i] = g(L[i]) = g(\sum_j w_j B_j[i]) = g(w^T b_j)$$

where w represents the combined filter weights and bias, and b the basis (feature) vector at pixel i with B 0 [i] = 1, and [·] the pixel index operator. Once learned they are typically fixed at inference time. When enough sparse depth measurements are available the weights w can instead be directly computed from data. Specifically, our weights are obtained through a least squares fit (LSF) from the bases B to the sparse depths S at valid pixels, which can then be used in place of the final convolutional layer. An overview of our proposed method is shown in Figure 10.

The objective function we wish to minimize for the least squares problem in matrix form is

$$\min_w \frac{1}{2} \sum_i g(w^T b_i) - s_i = \min_w \frac{1}{2} \|Bw - t\|$$

where s_i denotes an individual sparse depth measurement. The solution is the well-known Moore-Penrose pseudo-inverse which can be further regularized with parameter λ.

In practice, this problem is usually solved via lower-upper (LU) or Cholesky decomposition both of which are differentiable. Thus, the entire training process including our LSF module is differentiable which means

that it can be trained in an end-to-end manner. This is an important point since we have found that retraining the network with this fitting module produces much better results than simply adding the fitting procedure to a pretrained network without retraining. Effectively the retraining allows the network to make best use of the new adaptive fitting layer.

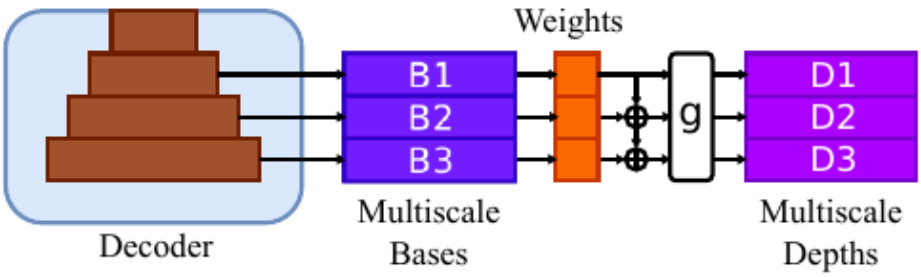

Figure 11: *Multi-scale depth prediction. The full resolution depth D 3 is reconstructed using all bases prediction.*

Self-supervised training formulates the learning problem as novel view synthesis, where the network-predicted depth is used to synthesize a target image from other view-points. Rather than predicting a depth map D' at each scale k separately, we propose to predict a set of bases B , as shown in Figure 11. Each of the basis vectors is obtained by upsampling features from corresponding scales in the decoder so the resulting basis images are band-limited by construction with coarser basis images corresponding to earlier layers in the decoder. The depth prediction at a particular scale s is then reconstructed using bases up to that scale. This technique greatly reduces various artifacts in the final depth prediction, but it still has one undesired property, With this formulation, predictions at different scales will work towards the same goal, which is to reconstruct the full resolution depth map. This approach is analogous to wavelet or Fourier encodings of an image where the basis maps are organized into band-limited components to represent the signal at various scales. Our LSF module handles this multi-scale approach quite naturally since we can simply allocate the basis maps in B among the desired scales, then upsample and group them back together to perform the fitting step. Henceforth we use this new multi-scale prediction scheme in all our experiments, even for supervised training where only the full resolution depth prediction is required.
We evaluate performance using standard metrics in the depth estimation literature. Note that for accuracy (δ threshold) we only report $\delta 1 < 1.25$, due to space limitations and the fact that the $\delta 2$ and $\delta 3$ are typically 99% for our experiments and thus provide limited insights. Following, we group results based on input modalities, where rgb denotes a network that only takes a color image as input. Figure 12 and Table 1 show quantitative comparisons between our proposed linear LSF module.

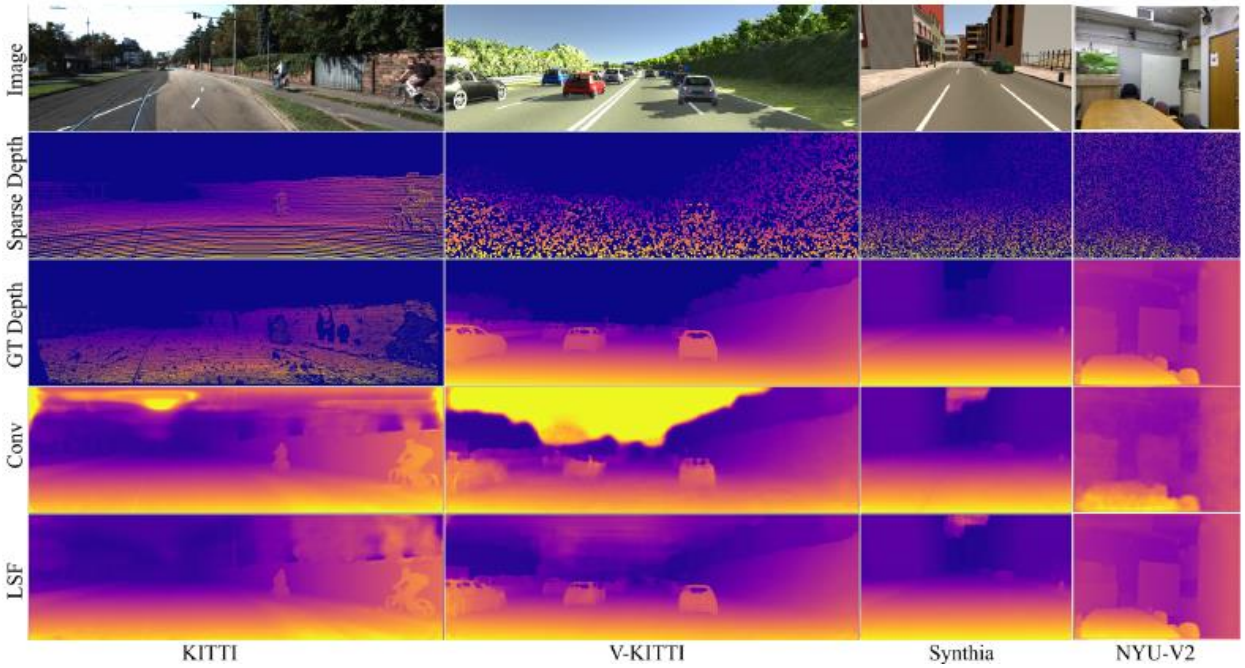

Figure 12: *Qualitative results on various datasets.*

Supervised Training			NYU-V2			V-KITTI			Synthia			KITTI		
Input	Method	Sparse	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1
rgb	conv	-	0.6244	0.8693	58.44	6.9998	14.653	66.43	2.3911	6.3915	76.09	1.8915	4.1164	86.24
rgb	pnp	0.2%	0.5517	0.6124	64.23	6.4701	13.990	70.18	2.1716	6.0084	81.37	1.6581	3.8019	88.67
rgb	lsf-	0.2%	0.4081	0.6124	77.86	5.8379	12.712	71.62	2.4089	6.2520	78.49	1.7033	3.5986	91.80
rgb	lsf	0.2%	**0.1826**	**0.3165**	**96.11**	**4.5122**	**9.7933**	**77.18**	**2.0104**	**5.6285**	**84.37**	**0.7716**	**2.0808**	**97.69**
(conv-lsf) / conv			+71%	+64%		+36%	+33%		+16%	+12%		+59%	+50%	
rgbd	conv	4%	0.1089	0.1679	99.20	1.5683	4.8982	94.71	0.7506	3.3322	96.50	0.3033	1.1392	99.57
rgbd	pnp	4%	0.1008	0.1604	99.24	1.5301	4.8798	94.81	0.7311	3.3217	96.60	0.2993	1.1343	99.57
rgbd	lsf-	4%	0.1127	0.1853	99.34	2.1049	6.1901	95.30	1.3220	4.6594	94.27	0.6319	2.2895	98.46
rgbd	lsf	4%	**0.0300**	**0.0735**	**99.83**	**1.2598**	**4.6227**	**97.43**	**0.5317**	**3.1146**	**97.85**	**0.2266**	**0.9988**	**99.67**
(conv-lsf) / conv			+72%	+56%		+20%	+6%		+29%	+7%		+25%	+12%	

Table 1: *Quantitative results on various datasets. conv denotes the baseline network, lsf indicates adding a linear LSF module to the pre-trained conv network without re-training for 5 iterations, and lsf is our linear fitting module (re-trained).*

We have been able to show that our approach is able to effectively learn a low dimensional manifold quite effectively. Moreover the experimental results show that this manifold can be learned quite quickly with more rapid convergence than competing approaches.

3.3 Improved Moving Target Detection on Mobile Computing Platform using Background Separation Technology as a Focus-of-Attention Mechanism:

During the Phase II effort, we developed a novel moving target detection and focus-of-attention mechanism for mobile platforms, that we named JanusNet [ZS+21] that combines several features

of the models discussed in previous sections. JanusNet is an efficient CNN model that can perform online background subtraction and robustly detect moving targets using resource-constrained computational hardware on-board unmanned aerial vehicles. It learns to extract and combine dense optical flow and generates a coarse foreground attention map using high-pass filters. It then combines optical flow, freground attention maps, and appearance features to separate foreground motion from background motion and to generate accurate pixel-wise masks of the moving objects in the scene. JanusNet addresses many limitations of the state-of-the-art discussed in earlier sections while also maintaining some of their most advantageous attributes. As opposed to many existing methods, the network does not make limiting assumptions about the sensor motion or the structure of the scene and is capable of operating in a variety of scenarios. It also does not require creation and maintenance of explicit background models, a memory and time-complex step. It leverages recent advances in deep learning techniques for robust estimation of dense optical flow from videos. This enables JanusNet to exploit both motion (dense optical flow) and image attributes (deep features) to identify independently moving objects from moving camera videos. The joint modeling of motion and appearance features for foreground segmentation has also recently been suggested in [CT+17, JXG17, RR+19], etc. However, most of these methods simply concatenate raw optical flow features and image features and pass them to convolutional layers that generate foreground segmentation. Our experiments have shown that while such models perform well on large and known objects, they do not perform well on previously unseen scenes/objects or disambiguating small objects as in the case of UAV videos. JanusNet tackles these challenges by using high-pass filters to generate multi-scale foreground attention maps and using a context layer that learns to combine these attention maps with raw optical flow, deep appearance features to improve results.

The following figure shows the neural network structure of JanusNet:

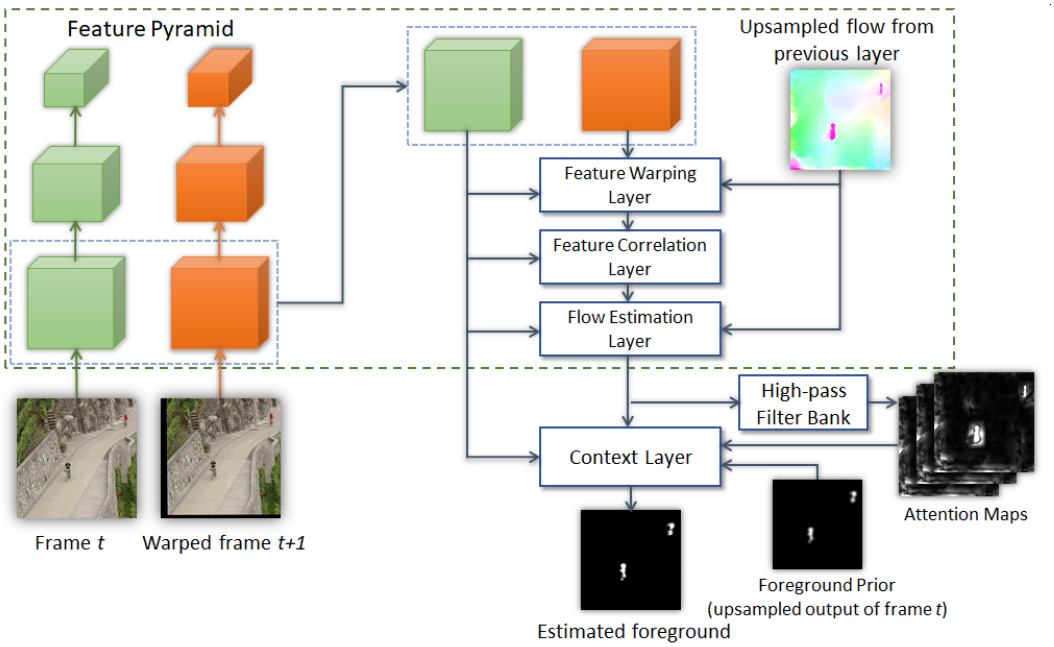

Figure 13: *JanusNet Network Structure*

Given two adjacent frames of a video, a sub-network first roughly estimates a global parametric motion between the two frames and produces roughly aligned frames using this estimate. The two aligned frames are used by the optical flow sub-network (that includes feature pyramid, warping, correlation, and flow estimation) similar to Pyramid Warping and Cost Volume (PWC)-Net to generate dense optical flow of the entire scene. The resulting estimated optical flow is then passed through multiple high-pass filters to highlight moving pixels. Finally, the context layer combines these highlighted pixels with image features, optical flow, as well as foreground priors (up-sampled foreground masks from previous frame) to fine tune the results based on semantic and contextual information. The network is trained end-to-end (except for the high-pass filters) using two training goals, one for optical flow generation, and another for foreground estimation.

3.3.1 Rough Global Motion Estimation

When the sensor is moving, the two adjacent video frames are not aligned with each other. Though the unaligned frames can be used directly for optical flow estimation and subsequent foreground separation, the large global motion between the frames may sometimes lead to poor optical flow performance, which in turn affects the quality of foreground. Therefore, we use a global motion estimator to roughly align the two frames to reduce global motion. Our global motion estimator follows a similar design as the deep homography model except that our model ingests higher resolution inputs and uses fewer layers and channels for the computational efficiency. In particular, the network uses 10 layers of convolutional layers with stride of 2 in every other layer. The first 4 layers have 32, 64, 96, and 128 channels respectively whereas the remaining 6 layers each have 128 channels. Finally, the network uses a linear layer to output the corresponding points pairs of the image corners to compute homography.

Compared with SIFT [L99] + Random Sample Consensus (RANSAC), ORB [RR+11] + RANSAC, a deep homography estimator is much faster but not as accurate (Table 2). However, in contrast to many existing approaches that assume a specific motion model and then use it as the basis of motion compensation and model generation, the type of motion model used (affine, homography, etc.) or the accuracy of estimation is not important here. Since the goal of this global motion estimation and alignment step is simply to reduce the effect of global motion on optical flow quality rather than completely eliminate the ego-motion from imagery, a rough alignment is sufficient for the model to produce accurate foreground segmentation (See Section 3.4).

Table 2: *Homography Transform Speed Comparison.*

Method	Resolution	Time/frame
SIFT + RANSAC	480x480	37.4ms
ORB + RANSAC	480x480	23.2ms
Original DeepH	resize to 128x128	7.5ms
Ours, small DeepH	resize to 320x320	5.2ms

3.3.2 Optical Flow Sub-Network:

The optical flow sub-network consists of feature pyramid extraction, warping, correlation, and flow estimation layers. The network follows a similar design as PWC-Net but trades accuracy for speed as discussed below:

Feature Pyramid Extraction Layer: For feature pyramid extraction, the network uses L-level pyramids consisting of convolutional layers (with images at 0^{th} level and the output of the deepest layer at the L^{th} level to extract image features at different scales. To extract features from input image I_t at level l: C_t^l, the network uses Residual Network (ResNet)-style convolutional layers to down-sample C_t^{l-1} by 2. In our experiments, we used L=4 levels with channels 16, 32, 64, and 96 respectively.

Feature Warping Layer: For each level, the network warps features of Image I_{t+1} towards Image I_t using the up-sampled flow from $(l+1)^{th}$ level O^{l+1}:

$$C_w^l(x) = C_2^l(x + up_2(O^{l+1})(x)) \qquad (1)$$

where x is the pixel position, $up_2(O^{l+1})$ is the upsampled optical flow from the l+1 th level.

Feature Correlation Layer: Given the features of image I_{t+1} and their warping towards I_t, the network calculates correlation scores for each pixel in image I_t with its corresponding neighboring pixels in warped image I'_{t+1}. We define the correlation score at level l $corr^l$ as:

$$corr^l(x_1, x_2) = \frac{1}{N^l}(C_1^l(x_1))^T(C_w^l(x_2)) \qquad (2)$$

where x_1, x_2 are pixel indices, T is the transpose operator, N^l is the number of channels for the l th-level feature pyramid. Calculating correlation between all possible combinations of pixel pairs x_1, x_2 takes too much computational resources, so we only compute correlations of pixel pairs in a $d \times d$ square centered at x_1. The time complexity of this module is $O(d^2 \times H \times W \times N)$, where H, W are the resolutions for C^l. At level l=1 where H and W are the large, this correlation layer becomes the speed bottleneck for our model, so we remove the correlation layer at level $l = 1$ for speed. Compared with PWC-Net, we reduce d from 9 to 5, and remove the lowest level's correlation layer for speed. Because we reduce d to 5, so our optical flow can only detect small motions, and that is one additional reason that we need to do a rough homography transform to reduce motion ranges before inputting images to our model.

Flow Estimation Layer: Flow estimation layer at level l consists of layers of Resnet-style convolutional layers which take concatenated correlation $corr^l$, features of first image C_1^l, and upsampled optical flow $up_2(O^{l+1})$ as inputs, and output estimated optical flow O^l. Compared to PWC-Net, we reduce the number of layers to 4 layers with channels 64, 32, 32, 2, where the final layer has 2 channels which correspond to horizontal and vertical pixel motions.

3.3.3 Attention Maps using High-pass Filter Bank:

As mentioned earlier, JanusNet exploits the optical flow of the scene from the flow estimation layer to separate foreground and background motions. As opposed to existing methods that use explicit motion factorization or subspace modeling for this task, JanusNet uses a convolutional model (learned in a supervised fashion) to identify independently moving objects. As we will show in Section 3.4, learning such a model directly from raw optical flow is extremely challenging given the large number of variables (scene structure, viewing geometry, etc.) that the output depends upon. Therefore, to guide this learning, JanusNet employs a bank of high-pass filters that attempts to provide attention to the regions where foreground motion is different from the background motion. The high-pass filter banks act as a focus-of-attention mechanism that highlight the regions that are more likely to contain independently moving objects, i.e., objects that exhibit different motion from their surroundings:

$$m_k^l = O^l - Avg(O^l, s_k^l) \qquad (3)$$

where m_k^l is the filtered optical flow at level l from high-pass filter k, and $Avg(O^l, s_k^l)$ is an average filter, with kernel size s_k^l at level l. Different values of s_k provide attention to the model at different scales (Figure 14).

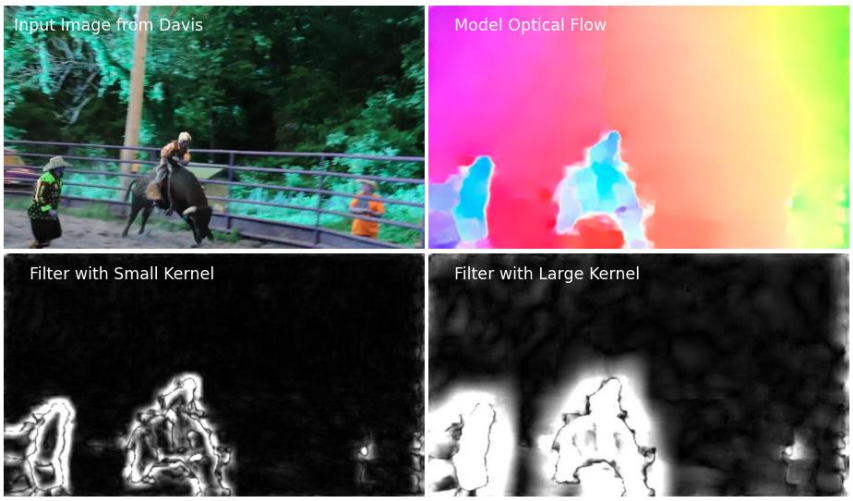

Figure 14: *Output of two high-pass filters with different kernel sizes*

In our experiments, we have found that k=2 with $s_1^l = 2^{l+1} + 1$ and $s_2^l = 2^{l+3} + 1$ provide sufficient coverage and accuracy for almost all practical scenarios, though additional filters can easily be added without any significant loss of performance. The network uses the output of each filter, m_k^l to generate attention maps (at different scales) as follows:

$$f_k^l(p) = min(1.0, \frac{|m_{k,x}^l(p)| + |m_{k,y}^l(p)|}{v}) \qquad (4)$$

$m_{k,x}^l, m_{k,y}^l$ are horizontal and vertical components of filtered optical flow m_k^l, f_k^l is the attention map of filter *k* at level *l*, *v* is the minimum motion of a pixel *p* to consider it as a moving pixel. If $\$\left|f_{k,x}^l(p) + f_{k,y}^{l}{}^{l(p)}\right| \geq v$, then $f_k^l(p) = 1.0$, else $f_k^l(p) < 1.0$. In our experiments, we used $v = 0.5$, i.e., the motion vectors smaller than 0.5 pixels at pyramid level 1=1 (1 pixel at image resolution) between 2 frames. For all practical purposes, vectors smaller than 0.5 pixels between two frames can be discarded without affecting the performance of the model.

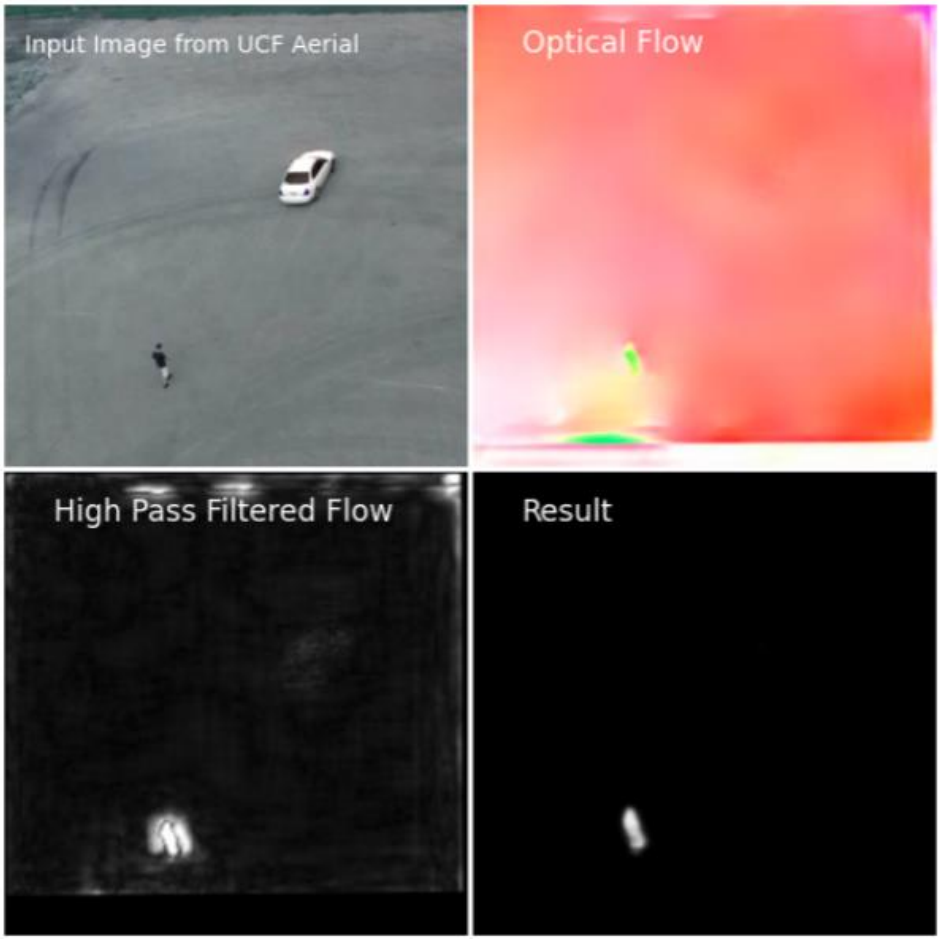

Figure 15: *Example of JanusNet output on University of Central Florida (UCF) Aerial Action Dataset. The person in the video is walking while the car is stationary.*

3.3.4 Context Layer:

The Context Layer produces the pixel-wise foreground mask by combining information from different sources that include: i) image features from video frames, ii) raw optical flow, iii)

attention maps of different scales from high-pass filter bank, and iv) foreground priors in the form of output from previous frame. We also updated the context layer to accommodate multi-scale attention maps from the filter banks.

$$fg^l = convs(C_1^l, corr^l, O^l, F_1, F_2, \ldots, F_k) \quad (5)$$

$$F_k = F(\gamma^{l+1}, f_k^l) = tanh(up_{t+1}(fg^{l+1}) \times f_k^l) \quad (6)$$

where f_g^l is the foreground segmentation at l[th] level, $convs(.)$ represents convolutional layers. In Eq 6, we multiply f_k^l with up-sampled foreground segmentation f_k^{l+1} to exclude motionless pixels. tanh is used to normalize the input of the CNN to range between -1 to 1.

Using supervised training, the context layer learns to combine and exploit semantics from images, motion cues from optical flow, attention maps, and priors from previous frames to determine object boundaries, remove shadows and spurious background objects, and identify pixels in the image belonging to independently moving objects.

3.3.6 Training the Network

The model is trained using two training goals: optical flow and foreground segmentation. Optical flow is trained with mean squared loss for each level with weights α^l; foreground segmentation is trained with binary cross entropy loss with the same weights α^l. In our experiments, we set α to be (0.1, 0.2, 0.4, 1.6) from level l=4 to l=1 respectively.

$$L_{flow} = \sum_l (\alpha^l \times MSE(O^l, O_{gt}^l)) \quad (7)$$

$$L_{fg} = \sum_l (\alpha^l \times BCE(sigmoid(fg^l), fg_{gt}^l)) \quad (8)$$

Figure 16: *Example of Janus Dataset with associated groundtruth.*

Due to lack of annotated datasets for background subtraction from moving camera, to train and validate the proposed model, we created Janus Dataset, a dataset of synthetic videos created in Unreal4 game engine. The dataset includes city, forest, beach, and castle scenes. We used various 3D models with animations and applied simple AIs to let them move randomly and intermittently. We set the camera above the ground and let it move and rotate randomly between frames. For each environment, we captured 10 to 20 videos, each with 200 frames, and each having different lighting effects, object positions, etc. We then used the Airsim package [SD+18] to generate the ground truth segmentation.

Since optical flow itself does not rely on the foreground segmentation, to provide a good initialization to our model, we first train the optical flow sub-network on the Flying Chairs [DF+15] dataset until the loss converges. Then we train one batch on optical flow using Flying Chairs dataset and then the second batch on foreground segmentation using Janus Dataset and so on. Janus Dataset is split into training and validation videos where the training videos include city, forest, and beach scenes, and the validation videos include the castle scene.

We used various data augmentation techniques: adding Gaussian noise, random rotation, random resize, random shift, random flip, artificial motion blurs, so that the trained model does not overfit on the relatively small Unreal4 training videos. To train the model, we used Adam optimizer with starting learning rate of 1e-4 and gradually decreased the learning rate. The training was continued until the learning rate reached 1e-8.

3.4 Testing and Evaluation:

We evaluated the JanusNet model both quantitatively and qualitatively over a number of standard datasets as well as inhouse datasets.

3.4.1 Quantitative Evaluation

We performed quantitative evaluation of the model and its different components using a labeled validation dataset. The results of the evaluation are shown in the table below:

Table 3: *Performance of JanusNet on a labeled validation dataset.*

Method	Precision	Recall
JanusNet	**0.767**	0.687
w/o Attention	0.704	0.643
NoHomography	0.663	0.661

Here, precision and recall are measured pixel-wise against ground truth. We also compared the updated network against several different state-of-the-art approaches on standard CDnet14 dataset, which is a widely used foreground segmentation dataset. Our updated network JanusNet, reached F-Measure of 0.56 (compared to the previously reported 0.53) which is similar to Gaussian

Mixture Model (GMM) [YM+18] and Kernel Density Estimation (KDE) [ED+02]. On the other hand, Foreground Segmentation Network version 2 (FgSegNetV2) [KSJ16] and Deep Convolutional Neural Network for Background Subtraction (DCBS) [BDR17] outperform our model on this dataset (see Table 4 below). However, JanusNet has several advantages over these approaches:

i) JanusNet is much smaller and faster;
ii) JanusNet is more general; FgSegNetV2 requires training on the exact scene. DCBS requires a corresponding background image, which is usually not available in practical scenarios;
iii) JanusNet works on both stationary and moving cameras. While FgSegNetV2 can handle some camera motion, it is limited in this capability as it must be trained on the same scene.

Table 4: *Comparison with the state of the art on CDNet14*

Method	Moving Camera?	Unseen Videos?	F-Measure
FrameDiff	False	True	0.21
GRASTA [HBS12]	False	True	0.36
JanusNet	**True**	**True**	0.56
GMM [YM+18]	False	True	0.57
KDE [ED+02]	False	True	0.57
KNN [YKJ19]	False	True	0.59
DCBS [BDR17]	False	Limited	0.76
FgSegV2 [KSJ16]	Limited	False	**0.97**

3.4.2 Results on UAV Videos

We also compare algorithms on UCF Aerial and Kaggle Drone videos datasets. To our knowledge, there is no benchmark UAV foreground segmentation datasets with ground truth labels, so we only compare the algorithms visually as shown in Fig 17. From the visual comparisons, our method outperforms existing algorithms on UAV videos (column 1-4) as our method successfully separates foregrounds and backgrounds while other algorithms detect too many false positives. Other algorithms have too many false positives because these algorithms are built upon the stationary camera assumption. FgSegNet, FgSegNetV2, and DCBS are not applicable for these videos because FgSegNet and FgSegNetV2 require labelled frames to train on these scenes first, and DCBS requires background images for these scenes, but both are not available. Our results on moving camera on these videos are significantly improved from the previously reported results as well. Those results (on the first 4 videos are shown in Figure 17).

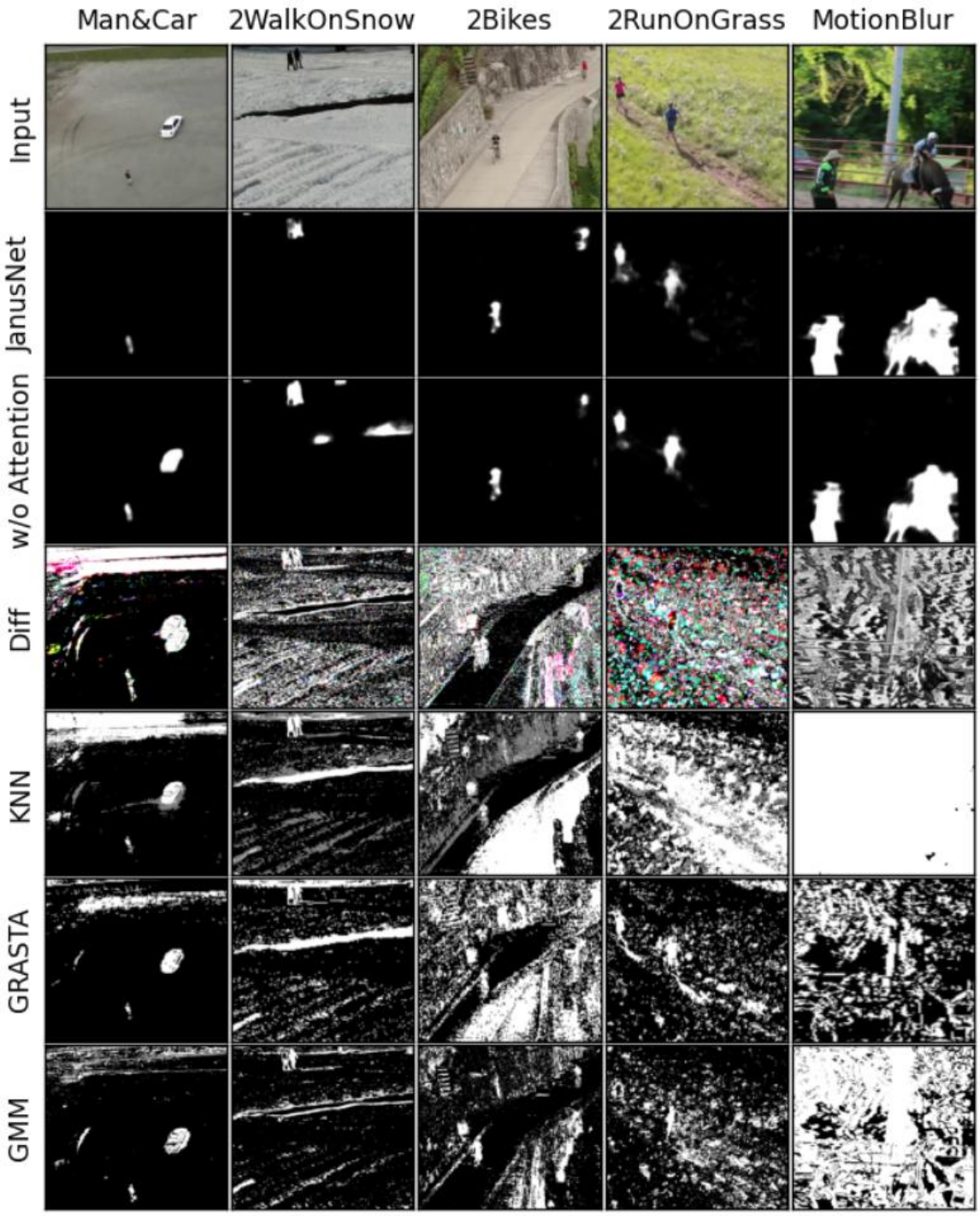

Figure 17: *Performance on various videos. Column 1 is from UCF Aerial Action, Column 2-4 are from Kaggle Drone Videos, Column 5 is from DAVIS Dataset. Column1-4 are videos from moving UAV cameras, the camera is also moving in the DAVIS dataset. You can see the camera movements from the "Diff" (frame difference) row.*

3.4.3 Qualitative Results on Stationary and Moving Camera Videos from Standard Datasets

We also tested the performance of the algorithm on new datasets containing videos from a variety of scenarios taken from both static and moving cameras. Some of the results from these datasets are shown below:

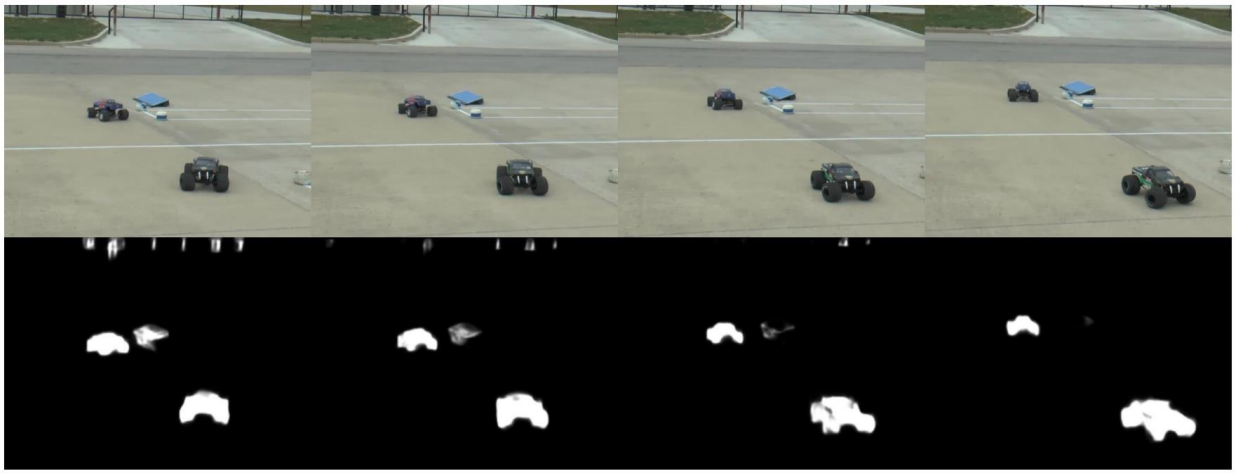

Figure 18: *Background-foreground separation results on a video from moving camera.*

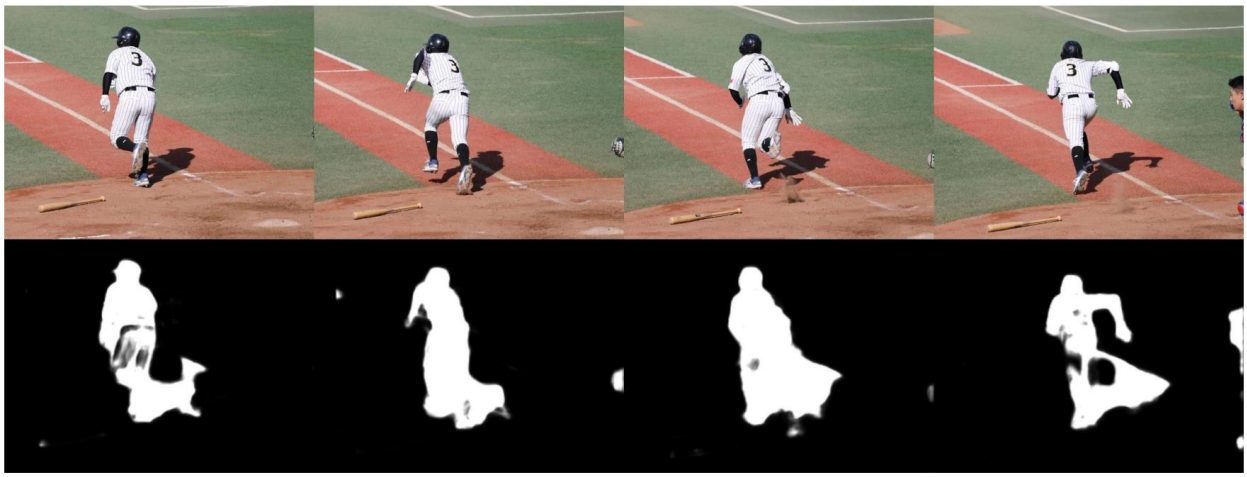

Figure 19: *Background-foreground separation results on a baseball video taken from moving camera.*

Figure 20: *Background-foreground separation results on a synthetic video from a simulated camera on a UAV platform.*

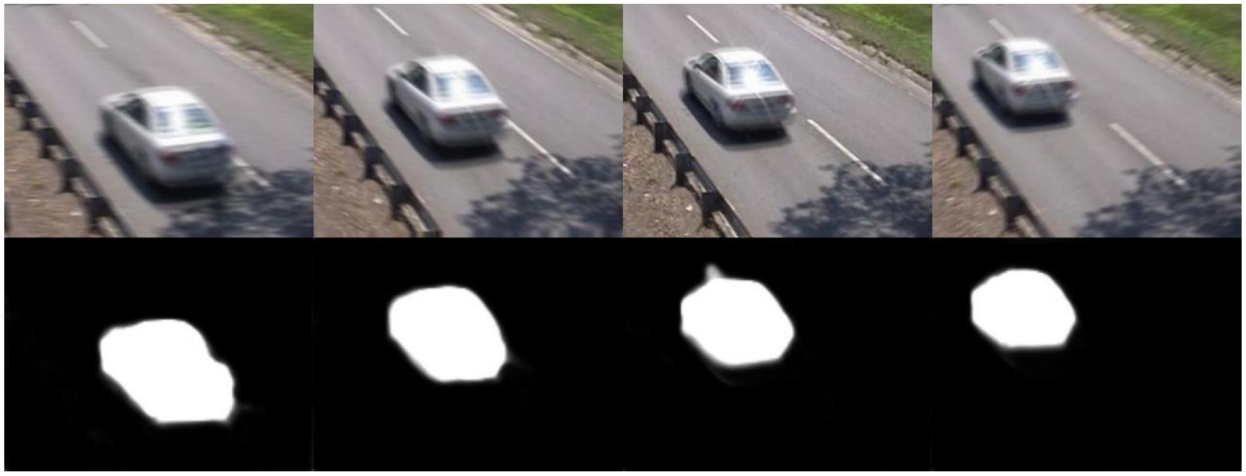

Figure 21: *Background-foreground separation results on a video from a UAV camera.*

Figure 22: *Background-foreground separation results on a pedestrian video from stationary ground camera.*

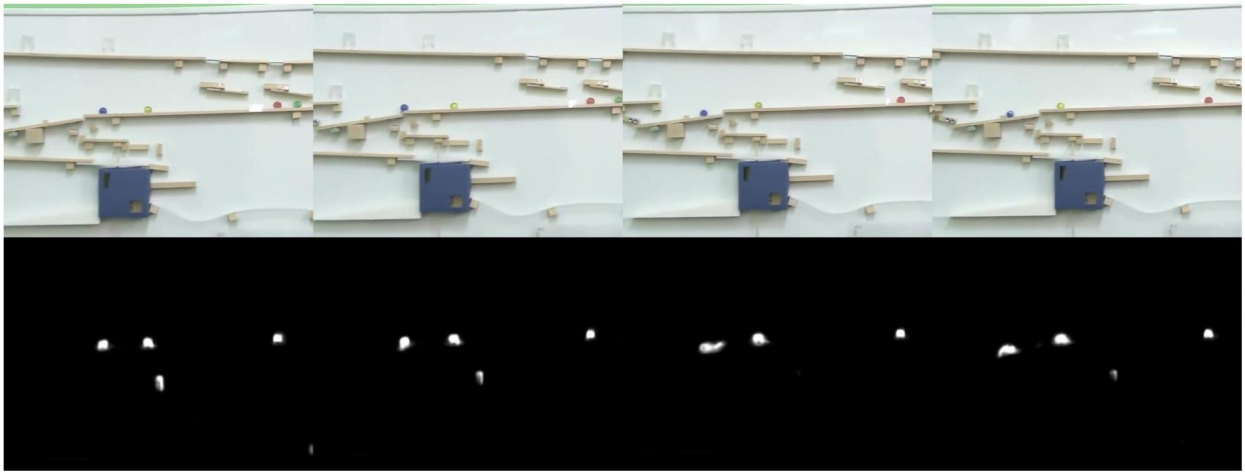

Figure 23: *Background-foreground separation results on a video from stationary camera where marbles are dropped from top to bottom.*

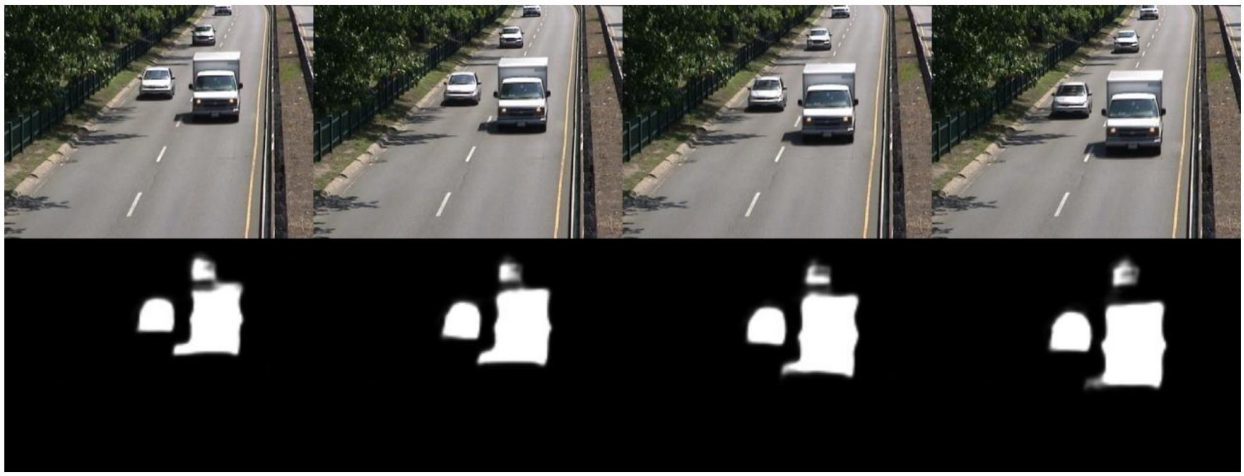

Figure 24: *Background-foreground separation results on a video from moving camera.*

Figure 25: *Background-foreground separation results on a traffic video from a stationary camera.*

Figure 26: *Background-foreground separation results on a video from stationary camera.*

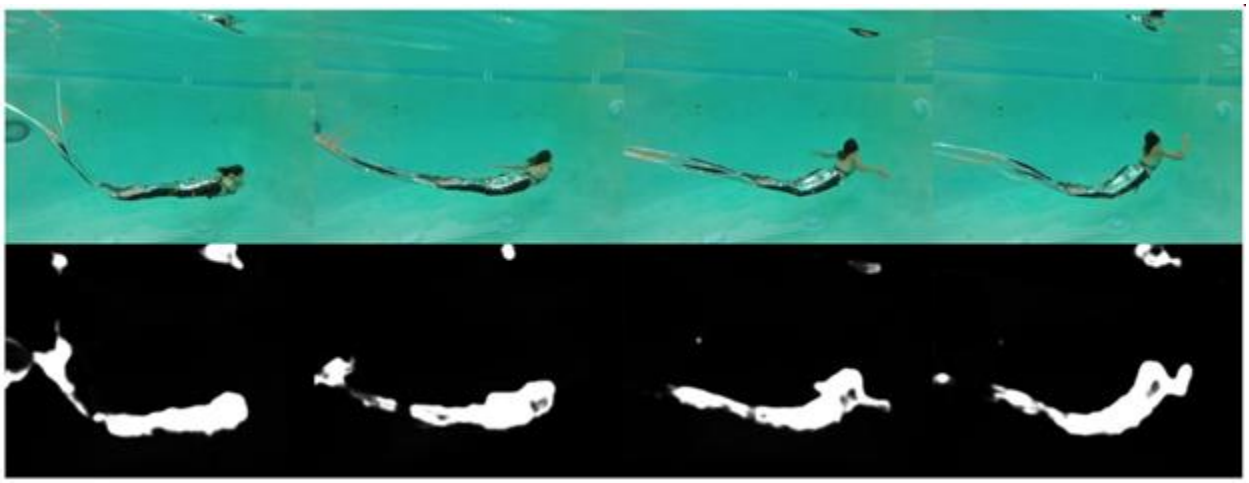

Figure 27: *Background-foreground separation results on a video of a swimmer from moving camera.*

3.4.4 Real-time Detection of Moving Pedestrian and Vehicles from In-house UAV Videos

We also tested the JanusNet system for the task of real-time detection of moving pedestrian and vehicles using UAV videos collected using our hex-rotor. Figure 28 and 29 show that the system is able to detect moving pedestrians and vehicles and provide focus-of-attention to the areas associated with these targets that enable advanced processing.

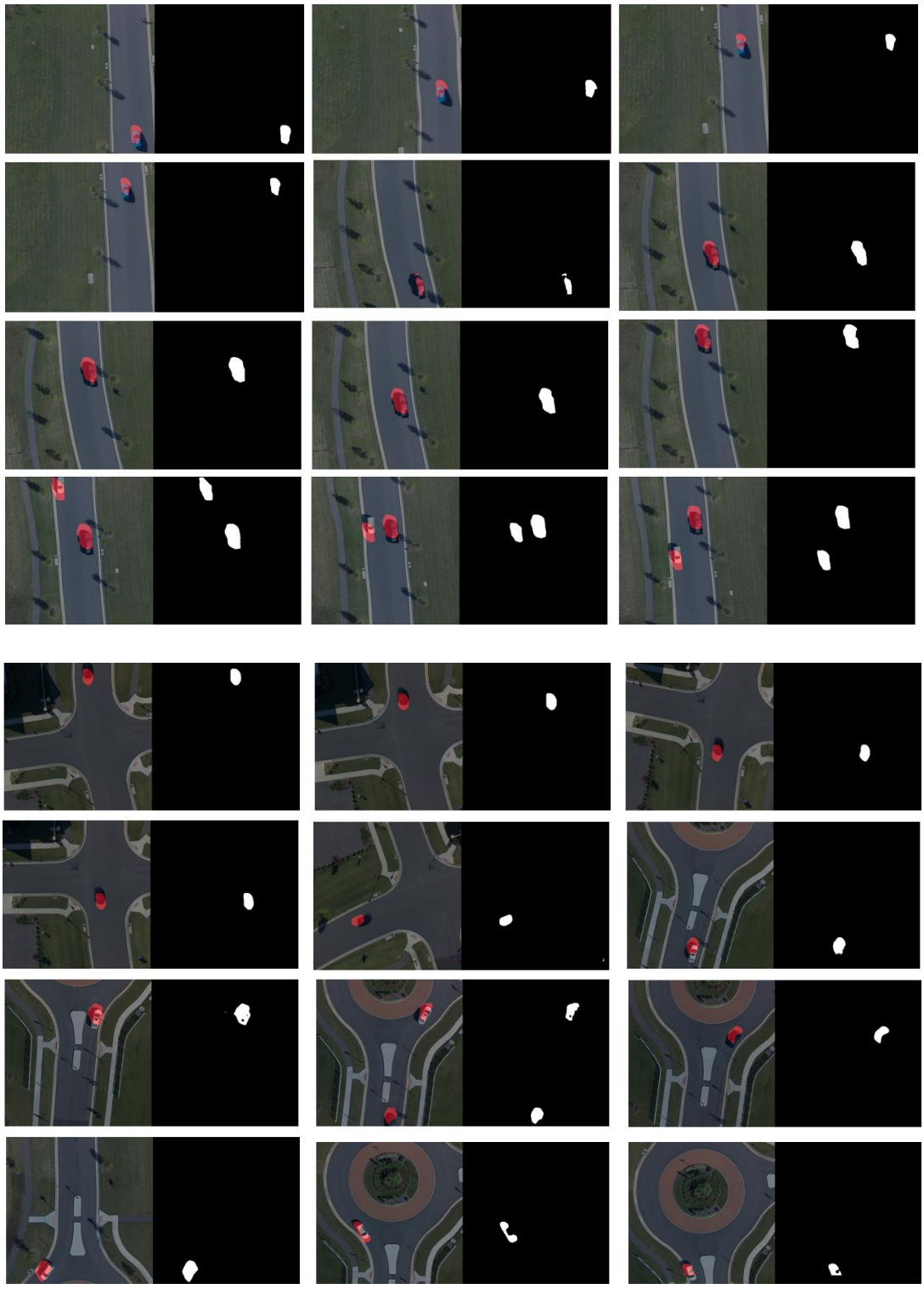

Figure 28: *Real-time detection of moving vehicles on two inhouse UAV videos.*

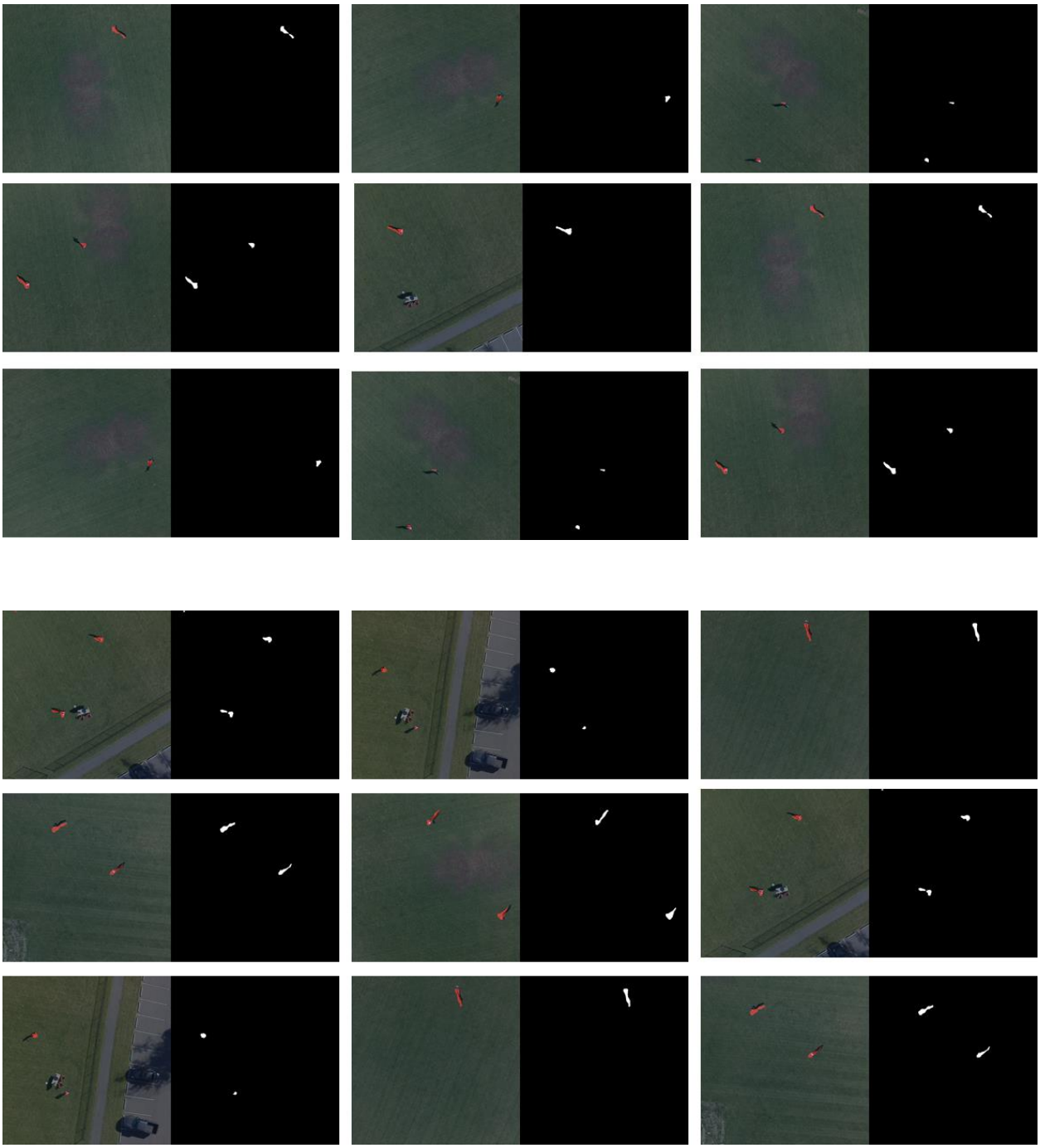

Figure 29: *Real-time detection of moving pedestrians on two inhouse UAV videos.*

3.5 Implementation of an End-to-End Software to Enable Robust Detection of Moving Targets in Videos:

During the Phase II effort, we ported the moving target detection technologies to an end-to-end software that was built using a Robotic Operating System (ROS)-based software architecture and optimized for real-time processing.

3.5.1 ROS-based Software Architecture

The software system is based on ROS, which provides a development and running platform for real-time systems. The functions of the system are implemented in a series of ROS nodes.

Real-time Computer Vision (RTCV) Processing Pipelines

The pipelines for RTCV processing are implemented based on ROS nodes. The ROS nodes can be configured to work together for different processing streams (Fig. 30), e.g. insert various processing nodes in the pipeline.

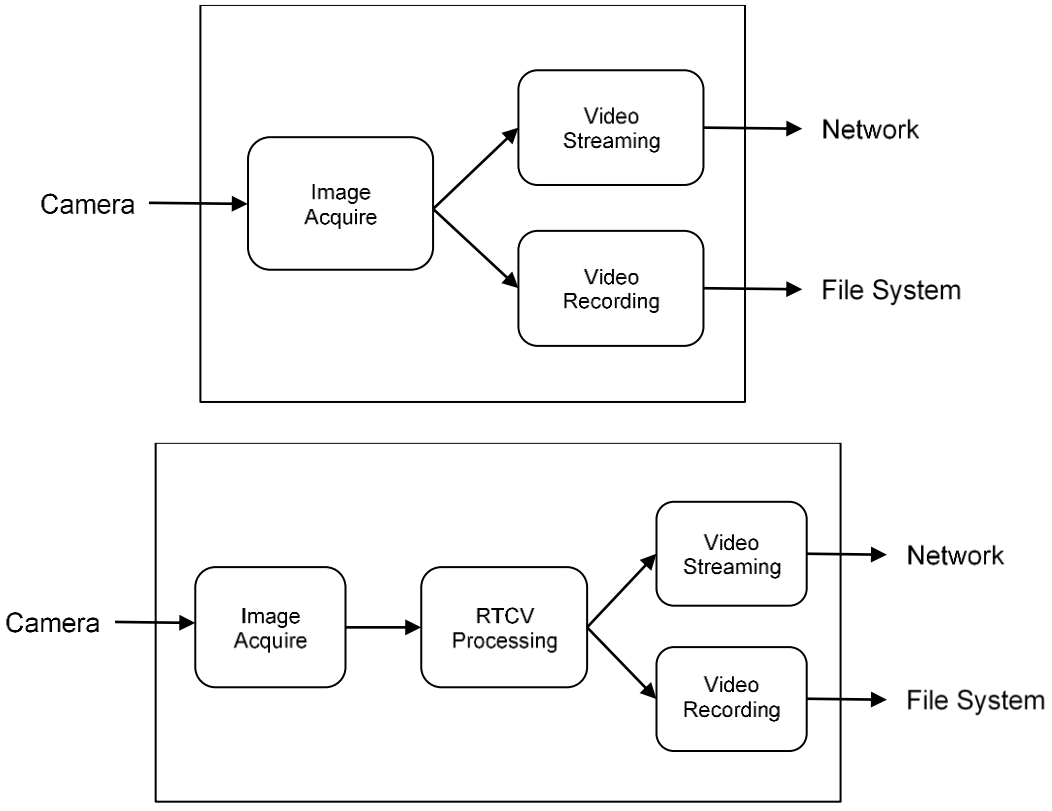

Figure 30: *RTCV processing pipelines*

Web-based Client

The client software of the system provides a real-time interface to access the status of the system, as well as the real-time images/video and the results of RTCV processing.

The web-based client is implemented based on the ROS bridge toolkit. A web server and a websocket server are deployed on the edge device as shown in Fig. 31. The client Graphical User Interface (GUI) is implemented by a set of web pages. The web pages communicate with ROS nodes through the websocket server.

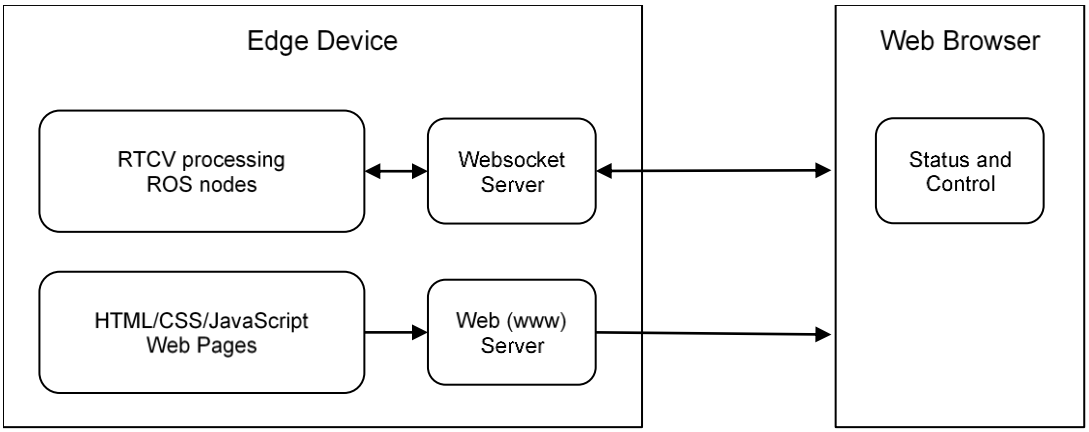

Figure 31: *Web based client for ROS system*

The system also supports real-time streaming of images/video. The web pages on the client side can display the video from cameras or the overlay images with RTCV processing results. The video stream is supported by two ways: (1) the RTCV processing nodes encode the images and send over RTP)/RTSP video stream, and (2) a web video server forwards images published by ROS topic over HTTP video stream (Fig. 32). The former is flexible and is able to process all intermediate images/video in the system (that may possibly be missed in a live onboard system), so is better for system debugging and testing. The latter can utilize dedicated hardware of Jetson Nano and therefore uses less CPU/GPU resource, which is better for run time system.

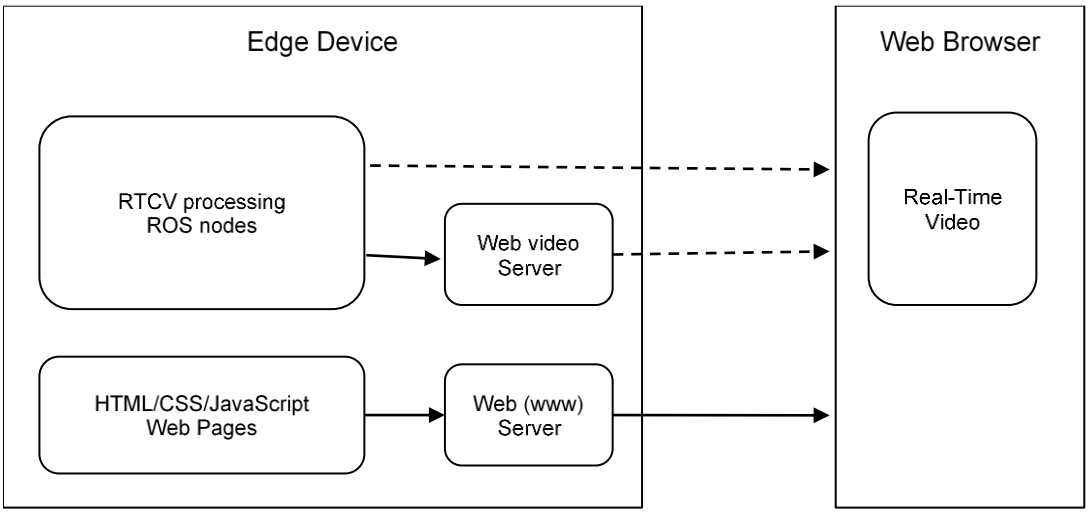

Figure 32: *Video stream from server to client*

Status Monitoring and Control

A ROS node is used to monitor the status of other working nodes. It generates an active report based on the diagnostic status or heartbeat from other nodes.

The working status of the ROS nodes could be controlled by dynamic reconfigure or specific ROS services. A ROS node is also used to run shell scripts upon request via ROS service.

The web-based client provides a user interface for status monitoring and control.

3.5.2 Real-Time Computer Vision (RTCV) Processing Pipelines

The software system is designed to support different RTCV processing pipelines to support different applications, by combining a set of ROS nodes.

Foreground Detection

The general purpose of foreground and background separation is to differentiate the pixels of the image to the foreground part and the background part. In practice, the foreground part is usually defined as the pixels that belong to the significantly moving objects. So, the foreground detection and moving objects detection are identical in some contexts.

The common processing stream for the foreground detection system is shown as Fig. 33. The original video from the camera and/or the overlay video with foreground mask (indicated by the dotted lines in Fig.33) can be recorded to the local file system or streamed to the client side.

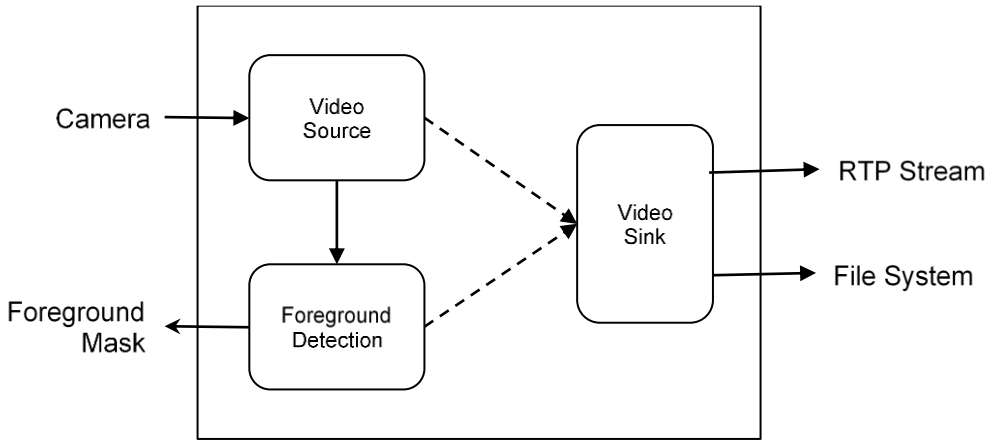

Figure 33: *Foreground detection pipeline*

The system supports different algorithms for foreground detection, for comparison and benchmarks. If the algorithm is implemented in C++, it can be loaded by a wrapper ROS node from external libraries (Fig. 34). If the algorithm is implemented in Python, it can be wrapped as a single ROS node (Fig.35).

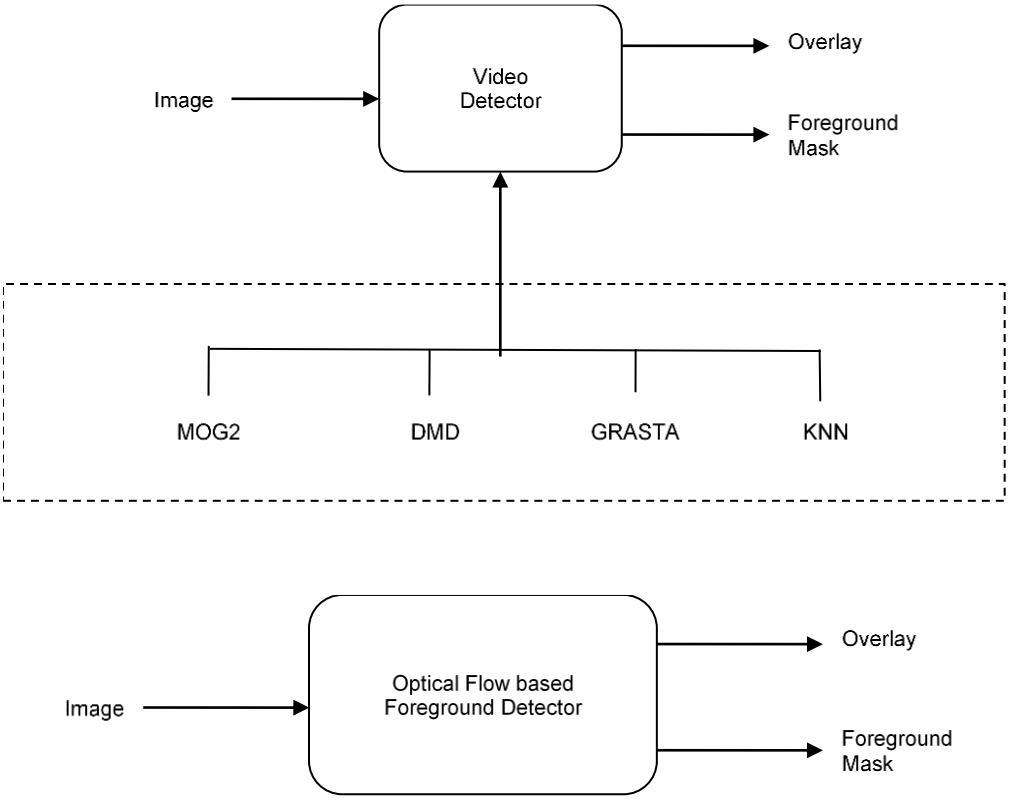

Figure 34: *Foreground detection algorithms*

3.5.3 Software Optimization

The aim of software optimization is to utilize the maximum capabilities of the hardware. The software is designed and optimized to utilize Jetson Nano dedicated hardware including the CPUs, GPU, Video Image Compositor (VIC), and Nvidia Encoder (NVENC).

OpenCV-based design

The basic system design is based on OpenCV. OpenCV provides software components to acquire images from the camera, write the images to file, or send the imagery to a network stream. By using GStreamer plugins from Nvidia as the backend, OpenCV can access Jetson Nano's CSI (Camera Serial Interface) camera and other dedicated hardware, as is shown below (Fig. 35).

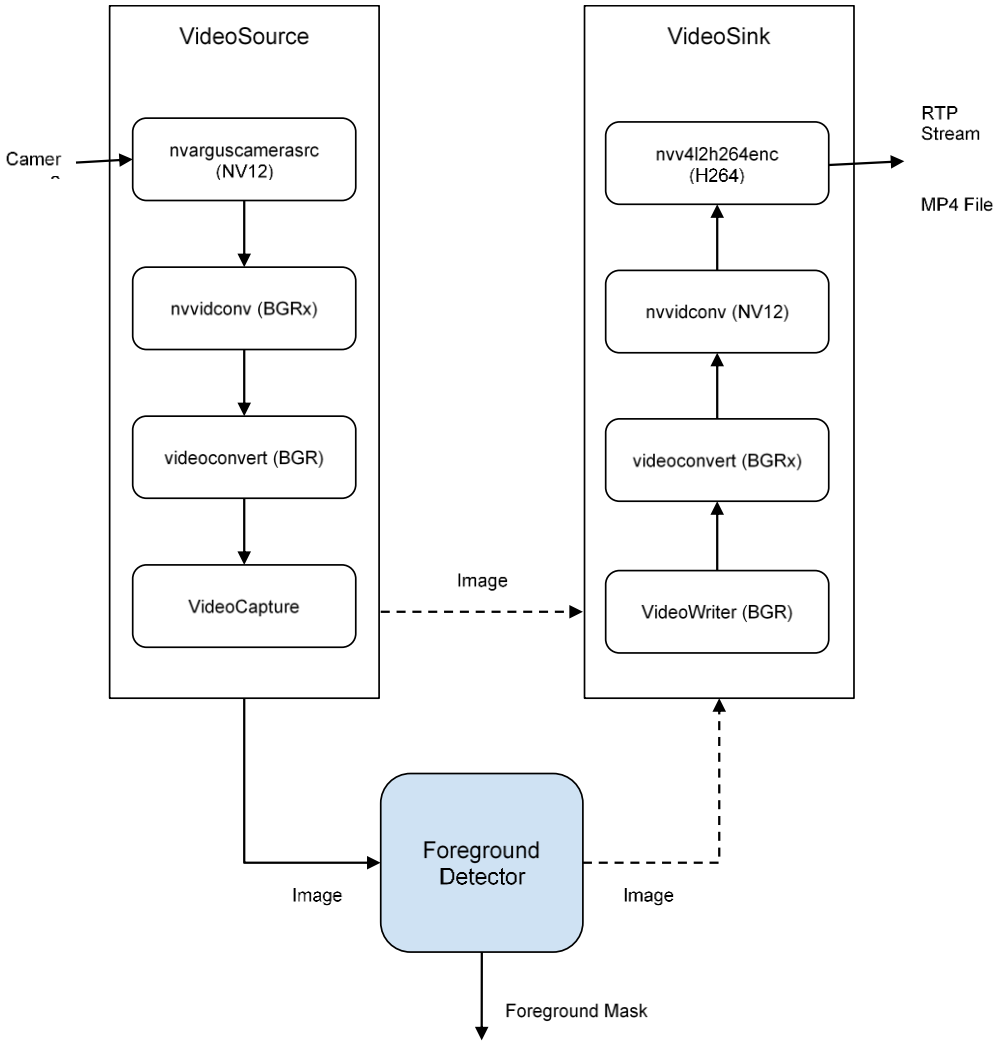

Figure 35: *OpenCV based ROS nodes for foreground detection*

In this design, the image processing stream is broken into three ROS nodes, the image is exchanged between ROS nodes using ROS topic. This design follows ROS convention and provides the most

flexibility to reconfigure and reuse the ROS nodes. The system can run on PC or Jetson Nano, working with live video from camera or video file.

DeepStream based design

Nvidia DeepStream SDK supports high performance image processing streams by accessing the dedicated Nvidia hardware (including Jetson Nano hardware). Following the DeepStream framework, a processing pipeline can be implemented in two ways: (1) create a library for the foreground detection algorithm and set it as a custom model for DeepStream ncinfer plugin (Fig. 36), and (2) create a custom DeepStream plugin to complete foreground detection (Fig. 37).

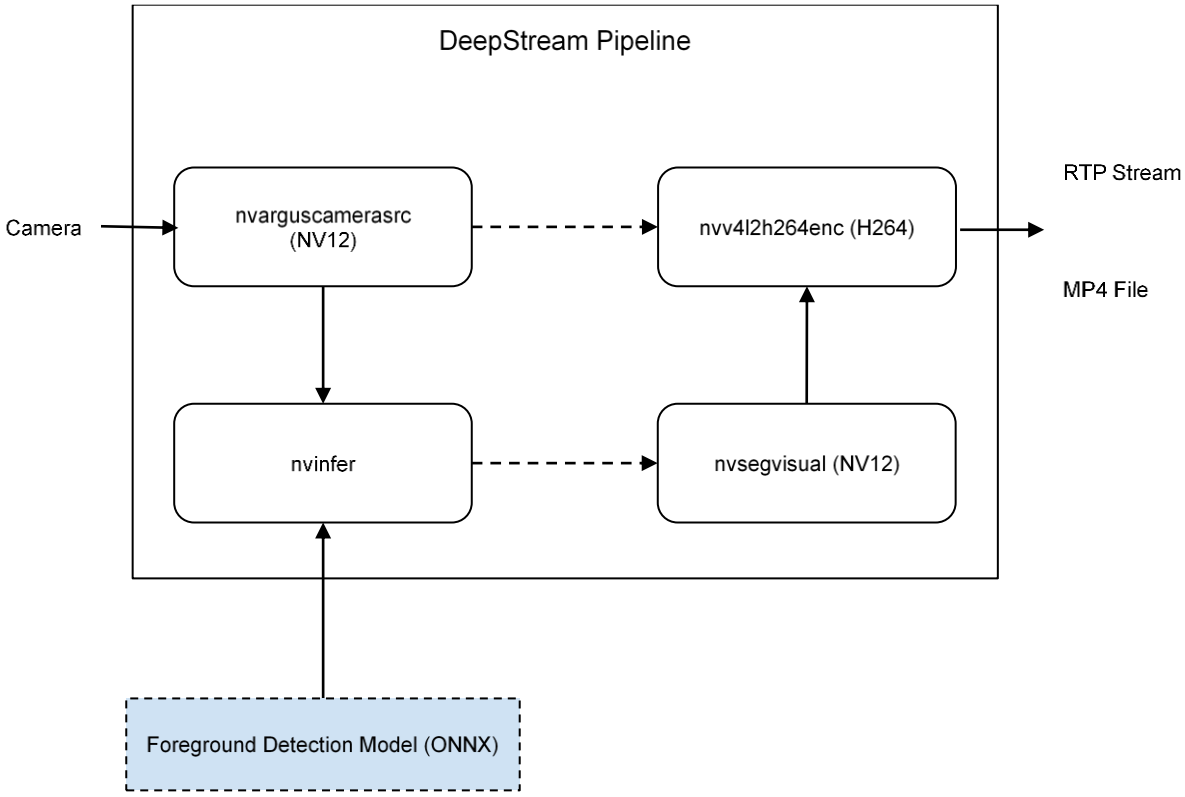

Figure 36: *Custom model for DeepStream pipeline*

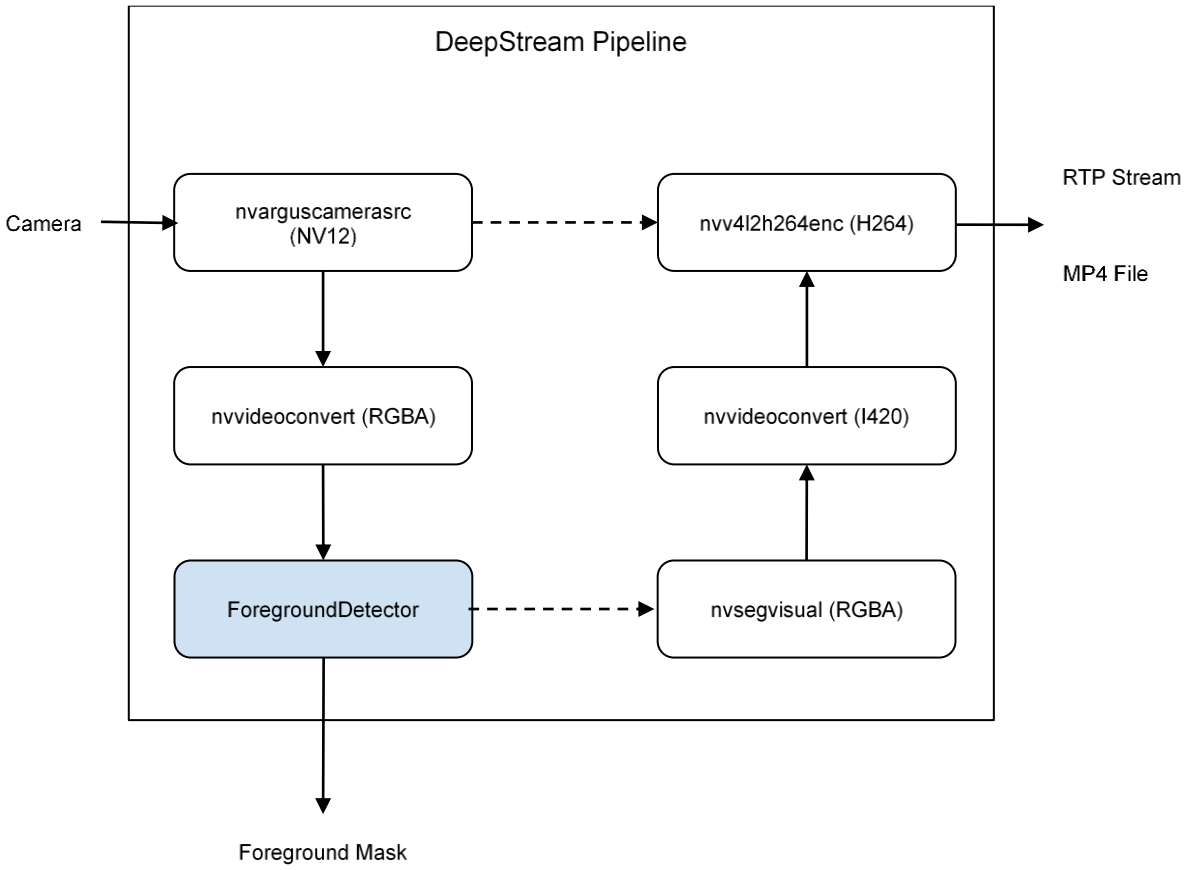

Figure 37: *Custom DeepStream plugin*

4. Conclusion

The Novateur Team has developed novel technologies for efficient background separation in both static and moving camera systems. The proposed background separation algorithms overcome several key limitations of the state-of-the-art methods that include: i) high computational demands, and ii) inability to handle sensor motion. These limitations of the traditional approaches for background separation severely reduce their applications in many modern applications. For example. Unmanned aerial vehicles (UAVs) are now widely used for both commercial and intelligence, surveillance, and reconnaissance (ISR) applications. Almost all of the existing UAV systems are equipped with one or more video sensors along with processing systems that can be used for onboard exploitation of video streams from sensors. However, the imagery from most of these sensors is either not automatically exploited at all or only exploited on the ground where more computational power is available for this task. Furthermore, a major factor in the recent UAV market growth is the proliferation of small-sized UAV platforms that are increasingly being developed for private, commercial and military use. These platforms have strict size, weight, and power (SWaP) limitations, that is, the onboard computations and sensor exploitation has to be performed on low-power and light-weight processing units. However, existing solutions are based on technologies that inherently require large computation resources. Typical solutions rely on 2D

mosaicking and 3D voxel-based modeling to handle platform motion that require power-hungry parallel processing units (GPUs, etc.) and do not lend themselves to small platforms regardless of software optimization techniques used. The proposed technology breaks new ground in efficient handling of platform motion under SWaP constrained environment by using novel sparse computational frameworks and low-rank subspaces. It enables automatic detection of targets-of-interest onboard the platform (with limited computational power) and therefore adds tremendous value to UAV systems.

5. References

[BDR17] Mohammadreza Babaee, Duc Tung Dinh, and Gerhard Rigoll. A deep convolutional neural network for background subtraction. arXiv preprint arXiv:1702.01731, 2017

[BJ03] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. IEEE TPAMI, 25(2):218–233, 2003.

[BYZ+18] C. Bi et al., "Dynamic Mode Decomposition Based Video Shot Detection," in IEEE Access, vol. 6, pp. 21397-21407, 2018.

[CT+17] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and MingHsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In Proceedings of the IEEE international conference on computer vision, pages 686–695, 2017

[DF+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[DVP18] M. Dimitrievski, P. Veelaert, and W. Philips, “Learning morphological operators for depth completion,” in International Conference on Advanced Concepts for Intelligent Vision Systems. Springer, 2018, pp. 450–461.

[ED+02] Ahmed Elgammal, Ramani Duraiswami, David Harwood, and Larry S Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proceedings of the IEEE, 90(7):1151–1163, 2002

[GJ+12] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, P. Ishwar, “Changedetection.net: a new change detection benchmark dataset,” in: Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on, 1–8, 2012.

[GK15] J. Grosek and J. Kutz, “Dynamic Mode Decomposition for Real-time Background/Foreground Separation in Video,” IEEE Transactions on Pattern Analysis and Machine Intelligence, to appear, 2015.

[GN+19] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, “Sparse and noisy lidar completion with rgb guidance and uncertainty,” arXiv preprint arXiv:1902.05356, 2019.

[HBS12] J. He, L. Balzano, and A. Szlam. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In Computer Vision and Pattern Recognition, June 2012.

[JXG17] S. D. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2117–2126, 2017.

[KFB15] J. N. Kutz, X. Fu, S. L. Brunton and N. B. Erichson, "Multi-resolution Dynamic Mode Decomposition for Foreground/Background Separation and Object Tracking," 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, 2015, pp. 921-929.

[KSJ16] L. Kurnianggoro, A. Shahbaz, and K. Jo. Dense optical flow in stabilized scenes for moving object detection from a moving camera. In 2016 16th International Conference on Control, Automation and Systems (ICCAS), pages 704–708, 2016

[L99] David G Lowe. Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. IEEE, 1999

[MCK18] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. arXiv:1807.00275, 2018.

[RR+11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An effcient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564–2571. IEEE, 2011

[RR+19] Hazem Rashed, Mohamed Ramzy, Victor Vaquero, Ahmad El Sallab, Ganesh Sistu, and Senthil Yogamani. Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019

[SD+18] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fdelity visual and physical simulation for autonomous vehicles. In Field and service robotics, pages 621–635. Springer, 2018

[TK+99] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, Wallflower: princples and practice of background maintenance, in: International Conference on Computer Vision (ICCV), 255–261, 1999.

[US+17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant CNNs. In 3DV, 2017

[YM+18] H. Yong, D. Meng, W. Zuo, and L. Zhang. Robust online matrix factorization for dynamic background subtraction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(7):1726–1740, 2018.

[YKJ19] Y. Yu, L. Kurnianggoro, and KH. Jo. Moving object detection for a moving camera based on global motion compensation and adaptive background model. International Journal of Control, Automation, and Systems, 17:1866–1874, 2019

[ZM+08] T. Zickler, S. Mallick, P. N. Belhumeur, D. Kriegman, "Color Subspaces as Photometric Invariants," International Journal of Computer Vision, 2008.

[ZS+21] Y. Zhao, K. Shafique, Z. Rasheed and M. Li, "JanusNet: Detection of Moving Objects from UAV Platforms," Workshop on Analysis of Aerial Motion Imagery (WAAMI), with ICCV 2021.