Investigation of moving objects through atmospheric turbulence from a non-stationary platform

Nicholas Ferrante[ab], Jérôme Gilles[b], and Shibin Parameswaran[a]

[a]Naval Information Warfare Center Pacific, 53560 Hull St, San Diego, CA, USA
[b]San Diego State University, 5500 Campanile Dr, San Diego, CA, USA

ABSTRACT

In this work, we extract the optical flow field corresponding to moving objects from an image sequence of a scene impacted by atmospheric turbulence *and* captured from a moving camera. Our procedure first computes the optical flow field and creates a motion model to compensate for the flow field induced by camera motion. After subtracting the motion model from the optical flow, we proceed with our previous work, Gilles et al,[1] where a spatial-temporal cartoon+texture inspired decomposition is performed on the motion-compensated flow field in order to separate flows corresponding to atmospheric turbulence and object motion. Finally, the geometric component is processed with the detection and tracking method and is compared against a ground truth. All of the sequences and code used in this work are open source and are available by contacting the authors.

Keywords: Target Detection, Atmospheric Turbulence, Camera Motion, Vector Field Decomposition, Wavelet Decomposition, Optical Flow

1. INTRODUCTION

In image processing, the detection and tracking of objects in an image sequence is a classic problem. Typically, the problem is broken into two steps: detection and tracking. The detection step has multiple variations in the literature: background subtraction,[2] temporal differencing,[3] optical flow,[4] and most recently the use of deep learning.[5] With the assumption of static camera, the method of background subtraction uses a running average of past frames to build and update a background model. When a new frame is introduced, the difference of the current frame and the foreground is assumed to correspond to real world motion. Optical flow based detection estimates a displacement vector field for each pixel in the image. The magnitude of this vector field is computed and any value above a certain threshold is assumed to correspond to a moving object. Neural networks exploit the relationship between object detection and image understanding in tracking high-level features within an image set. Once detected, a tracking algorithm is utilized in order to investigate the behavior of detections. The primary methods of tracking in image processing are point based tracking (*e.g.* Kalman[6] or particle filters[7]), kernel based tracking (*e.g.* Simple Template Matching,[8] Mean shift,[9] Support Vector Machine,[10] Layer based tracking[11]), and silhouette based tracking (contour and shape matching[12]). A comparison of these methods is presented by Balaji.[13]

In the case of a moving camera, there is an added complication of motion introduced through rigid body motion. In order to compensate for the false motion, a camera motion model is needed. One of the primary methods for creating a motion model is to use image registration between frames in order to find a relation across feature points. Feature points are correlated between frames and their displacements are then used to fit a model for camera motion. With the camera motion model, any locations in the image that do not conform to the predicted location are presumed to be moving objects. A review of object detection in video images captured by a moving camera is presented by Yaxdi.[14]

In this work, we add an extra level of difficulty by considering the presence of atmospheric turbulence in our image sequence. Atmospheric turbulence is typically due to an energy exchange within a fluid that separates

Further author information:
Nicholas Ferrante: E-mail: nicholas.ferrante1@navy.mil, Telephone: +1 619-553-7877
Jérôme Gilles: E-mail: jgilles@sdsu.edu, Telephone: +1 619-594-7240
Shibin Parameswaran: E-mail: paramesw@spawar.navy.mil, Telephone: +1 619-553-1554

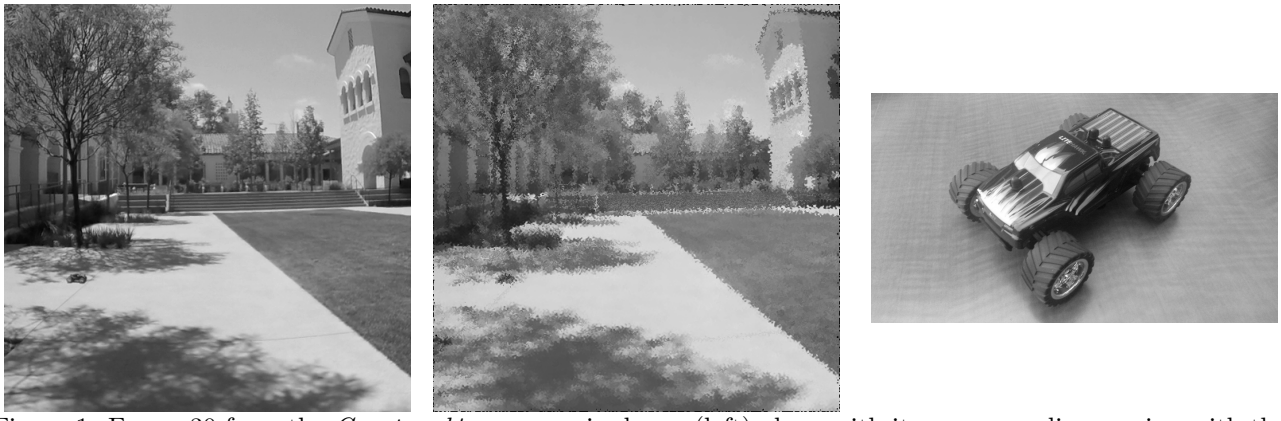


Figure 1: Frame 30 from the *Courtyard4* sequence is shown (left) along with its corresponding version with the introduction of simulated atmospheric turbulence (center). On the right is a close view of the remote control car used in each sequence.

an observer and an object. In our case, this fluid is air that is impacted by a temperature gradient, driving the change in the density of air and thus the change in the index of refraction. Through Snell's[15] law, the change in the index of refraction causes light to bend as it moves between a target and the observer.[16] This phenomenon introduces geometric deformation and blur within an image scene, bringing with it a myriad of problems in terms of image processing. With the temporal variance of the index of refraction, random oscillatory movement is introduced within the image scene that causes the false appearance of motion and degrade feature points needed for a camera motion model, therefore motivating a need for an object detection algorithm that may handle the combined impact of atmospheric turbulence as well as camera motion.

In this work, we introduce an optical flow based method that builds a camera motion model that is effective regardless of the impact of atmospheric turbulence. Once we build a camera motion model, we then are tasked with detecting moving objects within a camera motion compensated optical flow field. The proposed solution to this problem of detection is motivated by our prior work, Gilles *et al.*,[1] where a preprocessing method was introduced with the assumption of a static camera. In the proposed work, we extend the existing algorithm to detect and track objects from an optical flow field that contains atmospheric turbulence and global camera motion.

1.1 Data Collection

In order to properly test the methods proposed in this project, we needed a dataset that contains both atmospheric turbulence and global camera motion. Unfortunately, before the start of this project, there were no data sources available that contained small moving objects along with camera motion and atmospheric turbulence. Therefore, we created our own dataset to act as an addendum to the preexisting Open Turbulence Image Set (OTIS [*]).[17] In this section, we discuss the utilized equipment, methods of data processing, and some examples taken from the dataset. All sequences were recorded with a GoPro Hero 4 Black camera modified with a RibCage Air chassis, permitting the use of several lens types. A small tripod was used to hold the camera and it was manually rotated and translated in order to produce global camera motion. In order to introduce a moving object in the image sequence, a remote control car was utilized (see Figure 1). Data sequences without atmospheric turbulence were captured in the Engineering and Interdisciplinary Sciences Complex courtyard at San Diego State University. Data sequences with naturally forming atmospheric turbulence were taken at Peterson Gym 600 practice field, San Diego State University, during the afternoon hours on 24 July 2018.

Once the image sequences were captured, the MP4 files were downloaded onto a Macintosh computer, converted to PNG image files with the *ffmpeg* command[†], and cropped using the *imagemagick* crop command[‡] to contain a

[*] https://zenodo.org/communities/otis/

[†] https://ffmpeg.org/

[‡] http://www.imagemagick.org/

Table 1: Open source data set sequences from OTIS used in this research.

Name	Size	Turbulence
Courtyard1	$[512 \times 512 \times 100]$	Simulated
Courtyard2	$[350 \times 350 \times 113]$	Simulated
Courtyard3	$[300 \times 300 \times 138]$	Simulated
Courtyard4	$[512 \times 512 \times 100]$	Simulated
Field1	$[512 \times 512 \times 100]$	Natural
Field2	$[300 \times 300 \times 100]$	Natural
Field3	$[512 \times 512 \times 101]$	Natural
Field4	$[300 \times 300 \times 120]$	Natural
Field5	$[300 \times 300 \times 72]$	Natural

field of view with the moving object. For the sequences containing atmospheric turbulence, no further processing was needed, but for the sequences that were taken in the courtyard, an atmospherc turbulence simulator, developed by Tahtali, Fraser, and Lambert,[18] was implemented. The result from one frame is shown in Figure 1. It is important to notice that around the edges of the simulated sequence there exist black regions that may cause added complications during object detection. Therefore we recommend that the user crop the image edges in order to remove these artifacts. A summary of the datasets used in this work is shown in Table 1.
When assessing the effectiveness of our method, we require a ground truth for each of the sequences. Rather than manually observing the location of the car in each sequence, we coded a MATLAB application which provides a graphical user interface (GUI) to easily create a multiple-track ground truth which then can be used by the performance metric function.

1.2 Optical Flow Computation

The nature of this work does not lend itself well to detecting objects in the image domain, therefore we focus our efforts in the optical flow domain. The optical flow of an image sequence is the spatiotemporal vector field corresponding to the spatial movement of each pixel from one frame to the next. A general example of what we refer to by optical flow is demonstrated in Figure 2 where we have a region that moves from coordinates centered about (x_0, y_0) to new coordinates (x_1, y_1). As the optical flow field is dense within the image, we are unable to properly visualize the vectors in their traditional form, therefore we employ a technique which associates the direction of each vector with a color on the colorwheel from Figure 2 with the intensity of the color corresponding to the magnitude of the vector. There are many ways that optical flow may be computed, but for the purposes of this work we will primarily use the Horn-Schunck (HS)[19] and $TV - L^1$ algorithm.[20] These algorithms are both interfaced through MATLAB by using C code provided by the Image Processing OnLine (IPOL) journal.

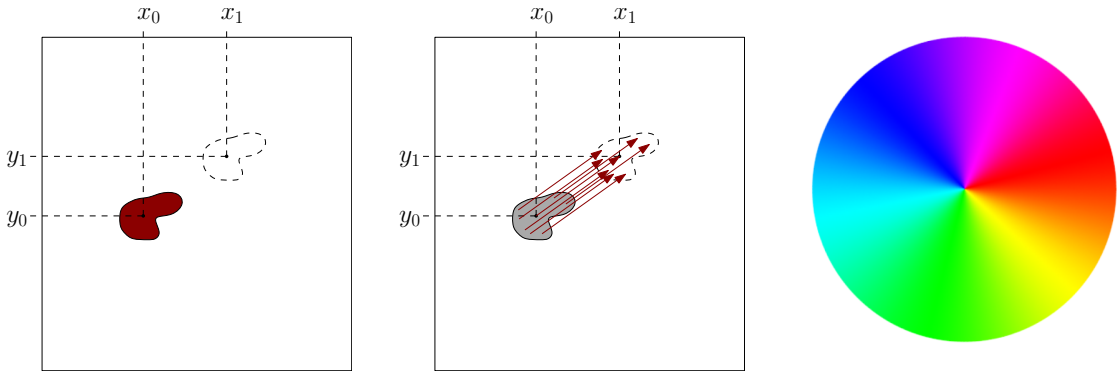


Figure 2: Illustration of the concept of optical flow. The left image represents an object (the red area) as its initial position centered around (x_0, y_0) and its position in the next frame (the dashed area) centered around (x_1, y_1). In the center figure, the set of arrows corresponds to the motion vectors *i.e.*optical flow. The colorwheel on the right provides the correspondence between color and movement direction.

2. GLOBAL CAMERA MOTION

For the purposes of this work, global camera motion or simply, "global motion", will refer to the rigid body motion that a camera may undergo. For generality, all angles of rotation and directions of translation are assumed to be possible. In this section, we shall discuss the physical model that governs the impact of global camera motion on perceived optical flow in the absence of atmospheric turbulence, followed by several methods that to build a model to approximate the camera motion flow. Finally, we provide a discussion on building a motion model when atmospheric turbulence is present.

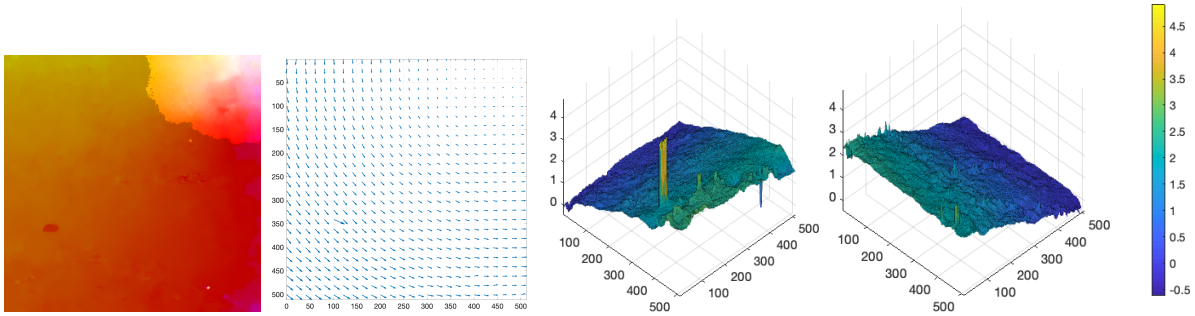
The impact of camera motion in an optical flow field is apparent when looking at a color representation of a flow in Figure 3a and the vector field quiver plot in Figure 3b. In order to investigate a model, we look to the x and y components of our optical flow field $\mathbf{V} = (V_x, V_y)$. Each vector component will produce a 2D surface, hence we utilize a surface plot in order to visualize the results as shown Figure 3c-3d. By investigating each component separately, we see that there are smooth underlying structures in the flow that we may later extract. It is also important to notice that the moving object stands out as a peak in the x component surface, thus any method that we choose must preserve the peak corresponding to the moving object. Through the use of the pinhole camera model,[21] we may derive the relation between real world coordinates $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ and the pixel coordinates (x, y) which upon differentiation in time yield a model for the optical flow field corresponding to camera motion. This relation relation is determined to be

$$\begin{aligned} V_x &= \frac{T_z x - T_x \mathfrak{f}}{\mathcal{Z}} + \omega_x \frac{xy}{\mathfrak{f}} - \omega_y \left(\mathfrak{f} + \frac{x^2}{\mathfrak{f}} \right) + \omega_z y \\ V_y &= \frac{T_z y - T_y \mathfrak{f}}{\mathcal{Z}} + \omega_x \left(\mathfrak{f} + \frac{y^2}{\mathfrak{f}} \right) - \omega_y \frac{xy}{\mathfrak{f}} - \omega_z x \end{aligned} \tag{1}$$

where $\mathfrak{f}$ is the focal length, $T_{(x,y,z)}$ and $\omega_{(x,y,z)}$ correspond to translation and rotation movement in the x, y, z direction. This model is derived in detail in Appendix A. Next, we present two methods which may extract the camera motion flow. First, we take a physics based approach by exploiting the optical flow camera motion model in Equation (1). Second, we employ a filtering method to robustly obtain a camera motion flow model.

2.1 Analytic Motion Model Derivation

First, we propose a method of determining each parameter from the camera motion model by making an assumption of constant depth $\mathcal{Z}$. Pseudo code for the algorithm proposed in this section is presented in Algorithm 1. By taking the curl of Equation (1) and asserting that $V_z = 0$ since we consider the focal length to be fixed, we



(a) Colorized Flow (b) Quiver plot (c) x componenet (d) y componenet

Figure 3: Representations of the optical flow field corresponding to the *Courtyard4* sequence at frame 30 without the inclusion of simulated turbulence.

get

$$\begin{aligned}\nabla \times \mathbf{V} =& \hat{\mathbf{i}}\left(-\partial_z V_y\right) - \hat{\mathbf{j}}\left(-\partial_z V_x\right) + \hat{\mathbf{k}}\left(\partial_x V_y - \partial_y V_x\right) \\ =& \hat{\mathbf{i}}\left(\frac{T_z y - T_y \mathfrak{f}}{\mathcal{Z}^2}\right) + \hat{\mathbf{j}}\left(\frac{T_x \mathfrak{f} - T_z x}{\mathcal{Z}^2}\right) \\ &+ \hat{\mathbf{k}}\left(-\omega_y \frac{y}{\mathfrak{f}} - \omega_x \frac{x}{\mathfrak{f}} - 2\omega_z\right). \end{aligned} \tag{2}$$

As the captured motion of interest is in the 2D image plane, we only need to consider the $\hat{\mathbf{k}}$ component, *i.e.*

$$(\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} = -\omega_y \frac{y}{\mathfrak{f}} - \omega_x \frac{x}{\mathfrak{f}} - 2\omega_z. \tag{3}$$

As mentioned in Appendix A, the image plane lies along the optical axis where its origin is at $(0, 0, \mathfrak{f})$. We note that the image domain is symmetric about the origin and ergo the average of the x and y coordinates is zero. Thus, denoting $\langle \cdot \rangle$ to be the spatial average value over the image domain we get,

$$\langle (\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \rangle = \langle -\omega_y \frac{y}{\mathfrak{f}} - \omega_x \frac{x}{\mathfrak{f}} - 2\omega_z \rangle = -2\omega_z, \tag{4}$$

which reveals

$$\omega_z = \frac{-1}{2} \langle (\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \rangle. \tag{5}$$

Next, taking the derivative with respect to each coordinate, we find

$$\begin{aligned} \partial_x \left((\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \right) &= -\frac{\omega_x}{\mathfrak{f}} \\ \partial_y \left((\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \right) &= -\frac{\omega_y}{\mathfrak{f}}. \end{aligned} \tag{6}$$

After taking the spatial average we obtain

$$\begin{aligned} \frac{\omega_x}{\mathfrak{f}} &= -\left\langle \partial_x \left((\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \right) \right\rangle \\ \frac{\omega_y}{\mathfrak{f}} &= -\left\langle \partial_y \left((\nabla \times \mathbf{V}) \cdot \hat{\mathbf{k}} \right) \right\rangle. \end{aligned} \tag{7}$$

This provides us all of the radial motion information. Next, we discuss translation components $T_{x,y,z}$. At this stage, we assume that the information from ω has already been computed. By taking the spatial average of the velocity field,

$$\begin{aligned} \langle V_x \rangle &= \langle V_x^T \rangle + \langle V_x^\omega \rangle = \langle V_x^T \rangle + \left\langle -\frac{\omega_y}{\mathfrak{f}} \left(\mathfrak{f}^2 + x^2 \right) \right\rangle \\ \langle V_y \rangle &= \langle V_y^T \rangle + \langle V_y^\omega \rangle = \langle V_y^T \rangle + \left\langle \frac{\omega_x}{\mathfrak{f}} \left(\mathfrak{f}^2 + y^2 \right) \right\rangle. \end{aligned} \tag{8}$$

where

$$\begin{aligned} \langle V_x^T \rangle &= \left\langle \frac{T_z x - T_x \mathfrak{f}}{\mathcal{Z}} \right\rangle = \left\langle \frac{T_z x}{\mathcal{Z}} \right\rangle - \left\langle \frac{T_x \mathfrak{f}}{\mathcal{Z}} \right\rangle \\ \langle V_x^T \rangle &= \left\langle \frac{T_z x - T_x \mathfrak{f}}{\mathcal{Z}} \right\rangle = \left\langle \frac{T_z y}{\mathcal{Z}} \right\rangle - \left\langle \frac{T_y \mathfrak{f}}{\mathcal{Z}} \right\rangle. \end{aligned} \tag{9}$$

Assuming $\mathcal{Z}$ to be constant, and hence $\frac{T_x}{\mathcal{Z}}$, $\frac{T_y}{\mathcal{Z}}$ and $\frac{T_z}{\mathcal{Z}}$ is also considered as constant values given by,

$$\begin{aligned} \langle V_x^T \rangle &= -\left\langle \frac{T_x \mathfrak{f}}{\mathcal{Z}} \right\rangle = -\frac{T_x \mathfrak{f}}{\mathcal{Z}} \\ \langle V_y^T \rangle &= -\left\langle \frac{T_y \mathfrak{f}}{\mathcal{Z}} \right\rangle = -\frac{T_y \mathfrak{f}}{\mathcal{Z}}, \end{aligned} \tag{10}$$

or equivalently,

$$\frac{T_x \mathfrak{f}}{\mathcal{Z}} = -\langle V_x^T \rangle = \left\langle -\frac{\omega_y}{\mathfrak{f}} \left(\mathfrak{f}^2 + x^2\right) \right\rangle - \langle V_x \rangle$$
$$\frac{T_y \mathfrak{f}}{\mathcal{Z}} = -\langle V_y^T \rangle = \left\langle \frac{\omega_x}{\mathfrak{f}} \left(\mathfrak{f}^2 + y^2\right) \right\rangle - \langle V_y \rangle. \tag{11}$$

Finally taking the divergence of our vector field we find the translation component T_z, *i.e.*

$$\nabla \cdot \mathbf{V} = 2\frac{T_z}{\mathcal{Z}} + 3\omega_x \frac{y}{\mathfrak{f}} - 3\omega_y \frac{x}{\mathfrak{f}} \tag{12}$$

and in spatial average,

$$\langle \nabla \cdot \mathbf{V} \rangle = 2 \left\langle \frac{T_z}{\mathcal{Z}} \right\rangle, \tag{13}$$

hence, again considering that $\frac{T_z}{\mathcal{Z}}$ is constant, we obtain

$$\frac{T_z}{\mathcal{Z}} = \frac{1}{2} \langle \nabla \cdot \mathbf{V} \rangle . \tag{14}$$

These equations give us a closed form solution to find all parameters within the model with the assumption of a constant $\mathcal{Z}$. From the numerical perspective, the derivatives are computed using a first order finite difference scheme. Prior to computing the derivative, the data is smoothed in order to mitigate error induced by localized fluctuations. The smoothing is performed with a 2D Gaussian filter on both V_x and V_y with a standard deviation equal to one seventh the number of columns and rows. The value of one seventh is chosen as it removes localized spatial fluctuations, such as moving objects and atmospheric turbulence, and may be changed for different applications.

Algorithm 1 Pseudo code corresponding to the analytical compensation model

1: Inputs: flow to decompose $\mathbf{V}$, focal length $\mathfrak{f}$
2: Smooth V_x and V_y with a Gaussian filter
3: Compute ω_z (5), and $\omega_{x,y}$ (7)
4: Using values $\frac{\omega_{x,y}}{\mathfrak{f}}$ compute $\frac{T_{x,y}\mathfrak{f}}{\mathcal{Z}}$ (11)
5: Compute $\frac{T_z}{\mathcal{Z}}$ (14)
6: Substitute the computed components into (1) to produce the approximated global motion flow $\mathbf{M}$.
7: Compute compensated optical flow $\mathbf{V}_c = \mathbf{V} - \mathbf{M}$
8: **return** $\mathbf{V}_c$

2.2 Empirical Camera Motion Model

In this section, we take an image processing standpoint to introduce a more robust empirical method to the camera motion flow as compared to the physically derived analytic method in the previous section. We proceed with motivation from Equation (1). Unfortunately, the assumption of a constant depth is not generally true, hence a more robust method is required.

Experimentally, when implementing a Gaussian filter as a preprocessing step for differentiation in the analytic method, the smoothed flow field produced appears qualitatively similar to the expected camera motion model that we aim to estimate. This result is expected as the assumption of a smooth function $\mathcal{Z}(x, y)$, the model from (25) will remain a smooth function even though it is no longer a simple polynomial. Recalling the end goal of this work, detecting moving objects in atmospheric turbulence, we have no need for the parameters of motion and thus we take the desired motion model to correspond to the output of a low pass filter. Taking a low pass filter will remove localized fluctuations in the optical flow resulting from atmospheric turbulence and object motion, thus leaving only the global camera motion. Just as we did in the analytic method, we empirically take the standard deviation of the Gaussian to be one seventh the number of rows and columns in order to mitigate the impact of localized fluctuations. The smoothing is performed separately on each component of the optical flow at each frame.

2.3 In the Presence of Atmospheric Turbulence

Since the main goal of this work is to detect objects in a flow field impacted by both atmospheric turbulence and global camera motion, we now need to look to the case where we include atmospheric turbulence. We will simulate atmospheric turbulence on the example from *Courtyard4* in Figure 3 so we may compare our result to the original sequence without atmospheric turbulence. In practice, the analytic method from Section 2.1 is unable to consistently account for camera motion observed in the sequence when turbulence is added as we loose the ability to take reliable derivatives due to the random motion induced from atmospheric turbulence. The empirical filtering method from Section 2.2 is able to handle the localized oscillations from the atmospheric turbulence by removing them through smoothing and hence is used routinely in this work. An example of the motion model is demonstrated in Figure 4 where we see that we indeed remove the motion that is introduced from camera motion as the resulting compensated optical flow is centered about zero. A colorized depiction of the flow field is shown in Figure 5 where we see the orange and red motion from the camera motion is removed leaving a primarily white background.

3. FLOW FIELD DECOMPOSITION

In this section, we address the problem of decomposing a 2D spatio-temporal vector field into its non-oscillating and oscillating components. We will proceed by using the empirical method described in Section 2.2 to remove the camera motion flow field as it produces the most consistent and reliable results. Moving forward, we refer to Figure 6 as our motivation for the proposed method. In this figure we see that, even in the presence of atmospheric turbulence, a moving object will correspond to a region of locally homogeneous vectors when we take a 2D+Time viewpoint on the problem. When viewed frame by frame, these homogeneous regions may not reveal a moving object, but using temporal information, these regions become apparent.

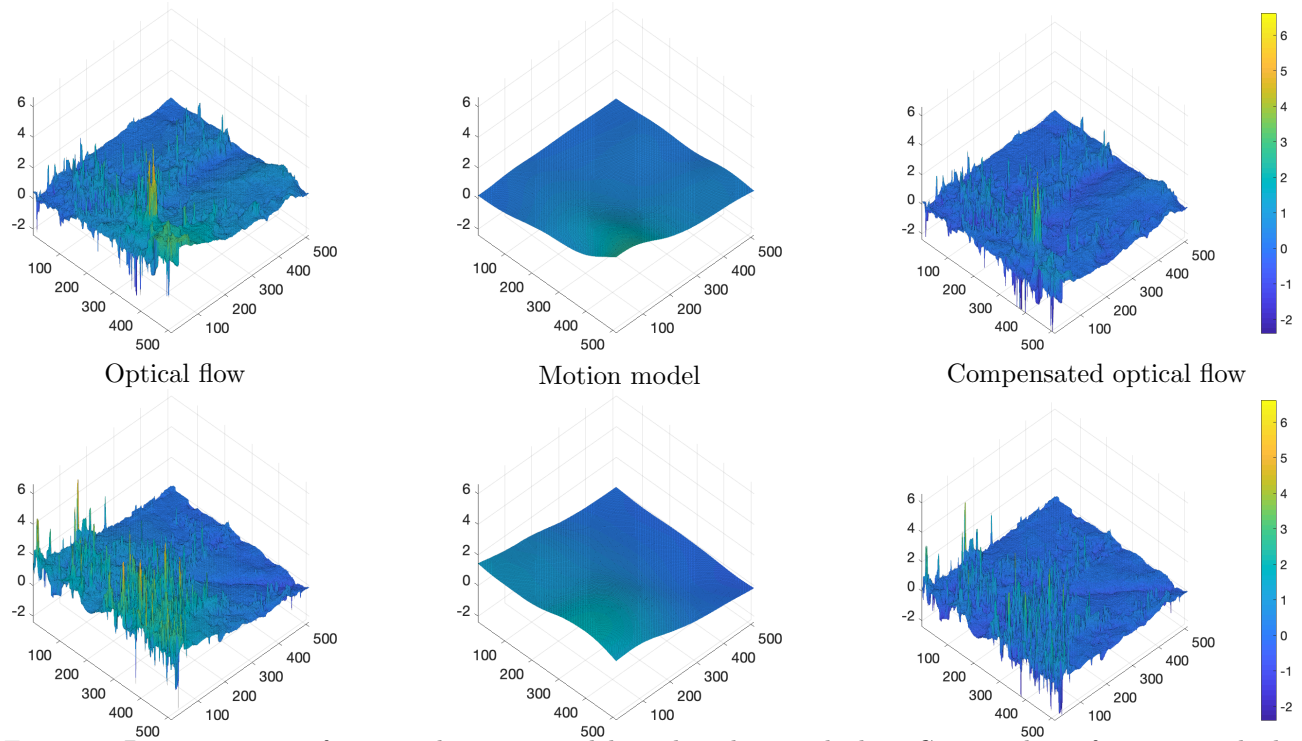
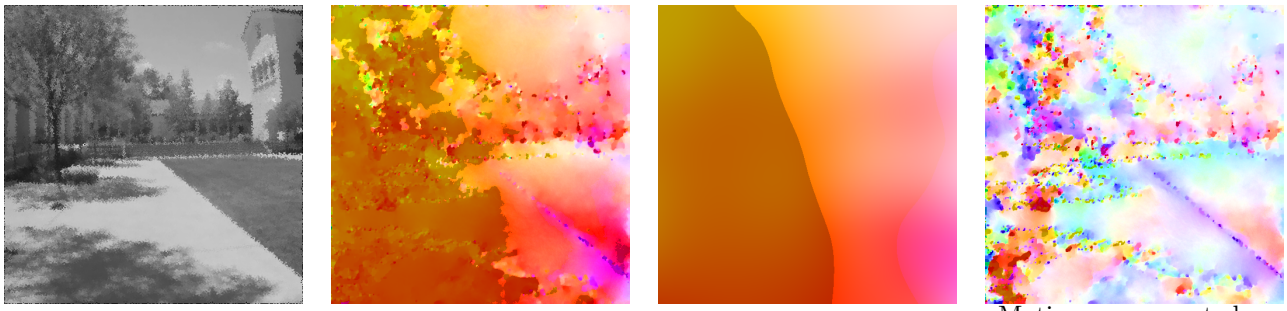


Figure 4: Demonstration of empirical motion model results when applied to *Courtyard4* at frame 30 with the inclusion of atmospheric turbulence. The x and y component are shown on the top and bottom left respectively. In the middle is the corresponding components of the camera motion model. The compensated optical flow is shown for each component to the right.



Frame 30 | Optical flow $\mathbf{V}$ | Camera motion model $\mathbf{M}$ | Motion-compensated optical flow $\mathbf{V_c}$

Figure 5: Demonstration of camera motion subtraction method from *Courtyard4* at frame 30 with simulated atmospheric turbulence. On the left, we see the image corresponding to frame 30 which has observable deformation due to simulated atmospheric turbulence. To its right, we see the corresponding optical flow where the impact of atmospheric turbulence is seen with the majority of the flow field occluded by camera motion. Next, we have the camera motion model that is computed by the empirical method. Finally, we have the camera motion-compensated optical flow field on the right, where we see only the impact of atmospheric turbulence.

3.1 Theoretical Background

A solution of this problem was proposed in our previous work[1] by using a decomposition model inspired by the classic "cartoon+textures" model used in image processing. The cartoon part corresponds to the geometric information while the texture part corresponds to the oscillating component. This decomposition was extended to 2D spatio-temporal vector fields, *i.e.* our camera motion compensated optical flow $\mathbf{V}_c = \big(V_{c_x}(x,y,t), V_{c_y}(x,y,t)\big)$. The idea is first to rewrite the vector field as a complex scalar field, *i.e.* $\tilde{V}(x,y,t) = V_{c_x}(x,y,t) + \imath V_{c_y}(x,y,t)$. Then we use both complex wavelet or curvelet transform[22] based on the computational or burden or resolution

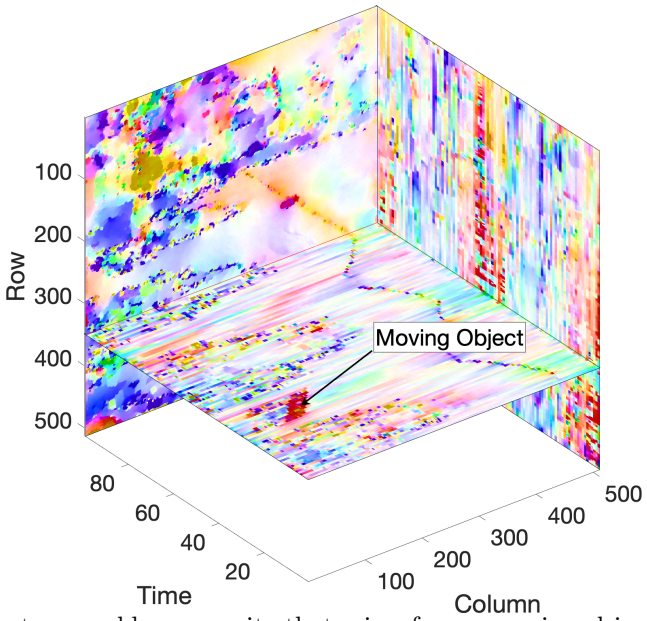


Figure 6: Visualization of the temporal homogeneity that arises from a moving object in the compensated optical flow from *Courtyard4* using the Slicer application.

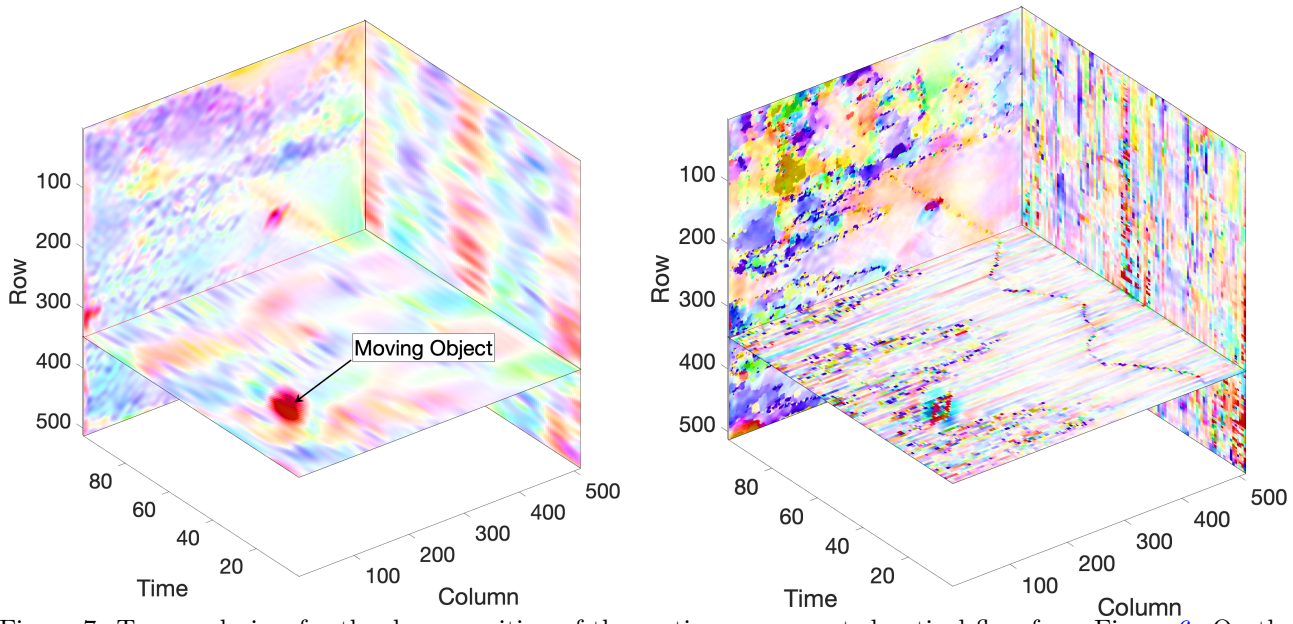


Figure 7: Temporal view for the decomposition of the motion compensated optical flow from Figure 6. On the left is the geometric component of the optical flow and on the right is the oscillatory component. It is visually apparent that the moving object stands out in the geometric component, while the atmospheric turbulence is left in the oscillatory component.

needs. The curvelet vector field decomposition model is given by

$$(\hat{u},\hat{v}) = \underset{\substack{u\in\dot{C}^1_{1,1}\\ v\in\dot{C}^\infty_{-1,\infty}}}{\arg\min} \|u\|_{\dot{C}^1_{1,1}} + J^*_C\left(\frac{v}{\mu}\right) + \frac{1}{2\lambda}\|I-(u+v)\|^2_{L^2}, \tag{15}$$

where $\dot{C}^1_{1,1}$ is a curvelet based function space and $\dot{C}^\infty_{-1,\infty}$ its dual. This model is utilized in order to preserve geometric information whereas changing to a wavelet based method will provide appreciably faster run time, as demonstrated in Figure 8, at the cost geometric resolution. The corresponding vector fields associated with $\hat{u}$ and $\hat{v}$ are respectively obtained by $\mathbf{u} = (\Re(\hat{u}), \Im(\hat{u}))$ and $\mathbf{v} = (\Re(\hat{v}), \Im(\hat{v}))$ where $\Re$ and $\Im$ denote real and imaginary components. If we denote

$$\forall z = |z|e^{\imath\theta} \in \mathbb{C}, \quad CShrink(z,\lambda) = \max(0,|z|-\lambda)e^{\imath\theta},$$

and $\mathcal{C}$ the curvelet transform, then the decomposition pseudocode is given by Algorithm 2.

Algorithm 2 Besov based decomposition numerical algorithm

1: Inputs: flow to decompose $\mathbf{V_c}$, parameters λ, μ, maximum number of iterations N_{max}
2: Initialization $n = 0, \tilde{V} = V_{c_x} + \imath V_{c_y}, u^0 = 0, v^0 = 0$
3: **repeat**
4: $v^{n+1} = \tilde{V} - u^n - \mathcal{C}^{-1}(CShrink(\mathcal{C}(\tilde{V}-u^n), 2\mu))$
5: $u^{n+1} = \mathcal{C}^{-1}(CShrink(\mathcal{C}(\tilde{V}-v^n), 2\lambda))$
6: **until** $\max(\|u^{n+1}-u^n\|_{L^2}, \|v^{n+1}-v^n\|_{L^2}) < 10^{-4}$ or $n = N_{max}$
7: **return** $\left(\Re(u^{n+1}), \Im(u^{n+1})\right), \left(\Re(v^{n+1}), \Im(v^{n+1})\right)$

3.2 Experimental Results

The decomposition algorithm was implemented in MATLAB and either uses the wavelet transform or the curvelet transform provided in the freely available Curvlab toolbox[§]. Experimentally, the choice of $\mu = \lambda = 1$ in the decomposition models works well for all sequences and is consistent with the results reported in.[1] The maximum number of iterations was fixed to five, though fewer iterations are observed to attain the threshold $\max(\|u^{n+1} - u^n\|_{L^2}, \|v^{n+1} - v^n\|_{L^2}) < 10^{-4}$. Before we decompose the flow field, we utilize a leave-one-out cross validation[23] on the magnitude of the motion compensated optical flow to determine frames containing outliers in the optical flow computation. As outlined in Algorithm 3, this method is performed by first computing the maximum magnitude of the flow at each frame in the sequence. Then, each maximum is removed from the data set and the change in the mean is observed. Any change that is outside of five-standard deviations of mean is considered as an outlier in the computation in the optical flow. Once the cross validation is performed any frames determined to be containing errors are linearly interpolated in time with neighboring frames deemed free of outliers. As errors in optical flow computation appear in regions with significantly larger magnitude, these regions act as Dirac-delta functions in the flow field. We know that the delta function is the neutral element in convolution and thus when using the wavelet decomposition the wavelet profile appears at the position of the outlier. This property leaves detection throughout the flow field impossible as the wavelet corresponding to the error is large enough in magnitude to be considered a valid moving object. The leave-one-out cross validation is performed with the function with $\kappa = 5$ as it produces consistent satisfactory results. Figure 7 illustrates the decomposition of the motion compensated optical flow from Figure 6 into its geometric and oscillatory components with the curvelet algorithm. Using this perspective, we qualitatively demonstrate the impact of our method. A quantitative analysis is left to Section 4 where we implement a detection and tracking algorithm. When the curvelet algorithm is implemented, it returns excellent results, but at a great computational cost. Taking note that Algorithm 2 is valid for both wavelet and curvelet transforms, we implement both methods in this work. We observe that implementing the wavelet transform gives similar qualitative results in the colorized optical flow as the curvelet transform, but a considerable drop in runtime is reached. In Figure 8, the speedup of the wavelet over the curvelet transform is emphasized with varying video resolutions. A quantitative comparison of the wavelet and curvelet tranform detection and tracking results are shown in Section 4 with quantitative metric scores. In general application, the use of wavelet or curvelet decomposition provide similar results, but if one wishes to maintain the most geometric information, the curvelet decomposition will provide the best results.

4. DETECTION & TRACKING

After the flow field decomposition is performed, the next procedure is the detection and tracking of the moving object. In order to detect a moving object in a flow field, we implement a velocity thresholding scheme as

[§] http://www.curvelet.org/software.html

Algorithm 3 Leave-one-out cross validation for outliers in optical flow

1: Inputs: flow $\mathbf{F}$, integer threshold κ
2: Initialization: Compute magnitude of the optical flow $\|\mathbf{F}_k\|$ at each frame $k = 1, 2, ..., N_{\text{frames}}$
3: **for** $i = 1 : N_{\text{frames}}$
4: Compute mean $M_i = \frac{1}{N_{\text{frames}}-1} \sum_{j \neq i} \|\mathbf{F}_j\|$
5: **end**
6: compute overall mean $\tilde{M}$ and standard deviation $\tilde{S}$ for $\tilde{M}$,

$$\tilde{M} = \frac{1}{N_{\text{frames}}} \sum_{l=1}^{N_{\text{frames}}} M_l, \quad \tilde{S} = \sqrt{\frac{1}{N_{\text{frames}} - 1} \sum_{m=1}^{N_{\text{frames}}} |\tilde{M} - M_m|}$$

7: inds $\leftarrow \left\{ i : \left| M_i - \tilde{M} \right| > \kappa \tilde{S} \right\}$
8: **return** inds

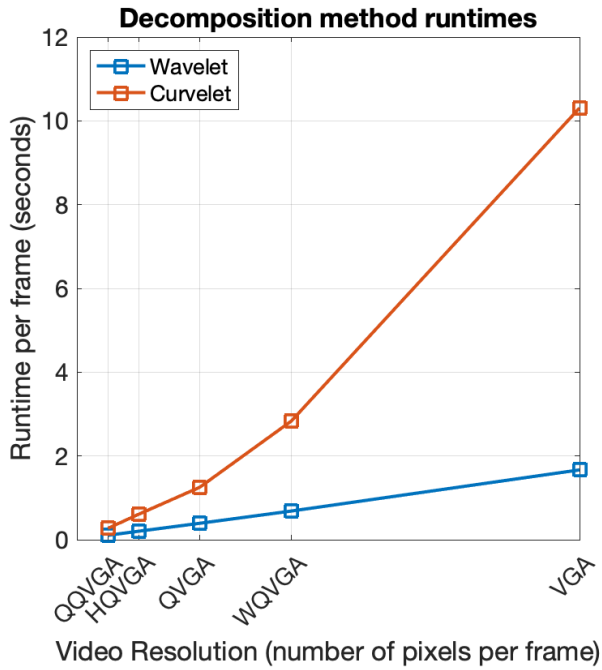


Figure 8: Comparison of a per-frame run time for the wavelet and curvelet decomposition for varying standard video sizes.

explained in Section 4.1. Once the detection step is completed, the tracking step is implemented via a Kalman Filter.[24]

4.1 Object Detection

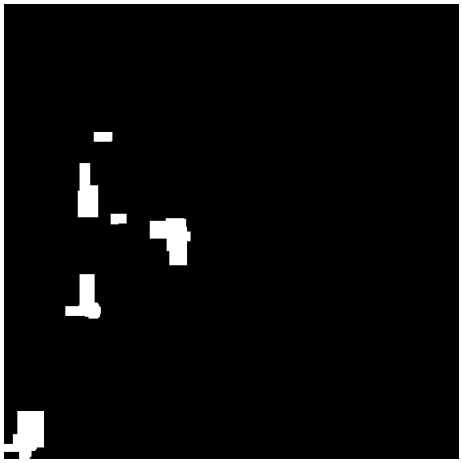
In order to detect moving objects in the geometric component of our flow field decomposition, we need to recall that parts of the low frequency oscillations from the atmospheric turbulence will remain in the geometric component. With this thought in mind, the remaining low frequency oscillations captured in the geometric component will not correspond to large values (typically close to zero), but will be detected if a naive threshold operation is performed. We see from a surface plot of the data and a color representation of the flow in Figure 9, that the moving object stands out from its surroundings in the geometric component of the flow than in the motion compensated optical flow. In order to determine a threshold value, we compute a mean and a standard deviation of a set number of past frames (typically set to five), then set the threshold to be five standard deviations away from the mean value. Implementing this procedure, we have the threshold value T_i at each frame i, to be $T_i = \mathfrak{M}_i + 5\sigma_i$ where $\mathfrak{M}_i$ is the mean magnitude and σ_i the standard deviation of the magnitude of a flow $\mathbf{F}$. In practice we compute $\mathfrak{M}_i$ and σ_i through

$$\mathfrak{M}_i = \frac{1}{N_{\text{rows}} N_{\text{cols}}} \sum_{r=1}^{N_{\text{rows}}} \sum_{c=1}^{N_{\text{cols}}} \|\mathbf{F}(r,c,i)\|, \quad \sigma_i = \sqrt{\frac{1}{N_{\text{rows}} N_{\text{cols}} - 1} \sum_{r=1}^{N_{\text{rows}}} \sum_{c=1}^{N_{\text{cols}}} |\|\mathbf{F}(r,c,i)\| - \mathfrak{M}_i|}.$$

where N_{rows} and N_{cols} are the number of rows and columns in a frame, respectively. Once the detection method is implemented, mathematical morphology operations are applied on the mask through image opening and closing. Finally, the centroid and bounding box of each region are computed in the mask.

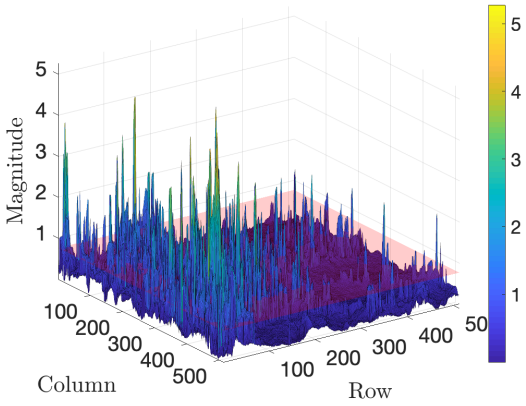
4.2 Object Tracking

With centroids in hand, the detected locations are assigned to existing tracks through the Hungarian assignment Algorithm[25] using the optimized algorithm from Miller, Stone and Cox.[26] Once the detections are assigned to tracks, a prediction is made for each track by way of a Kalman Filter.[24] In order to assign detections to tracks,



Motion Compensated Optical Flow

Geometric Component



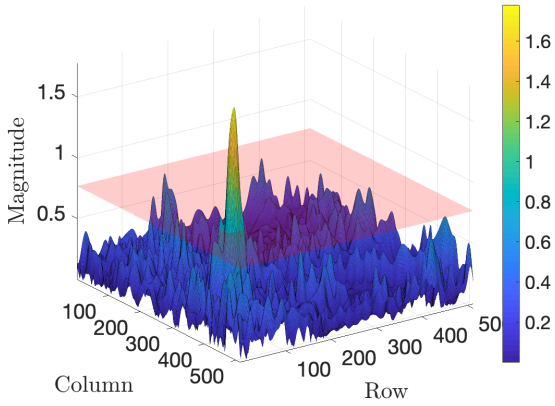
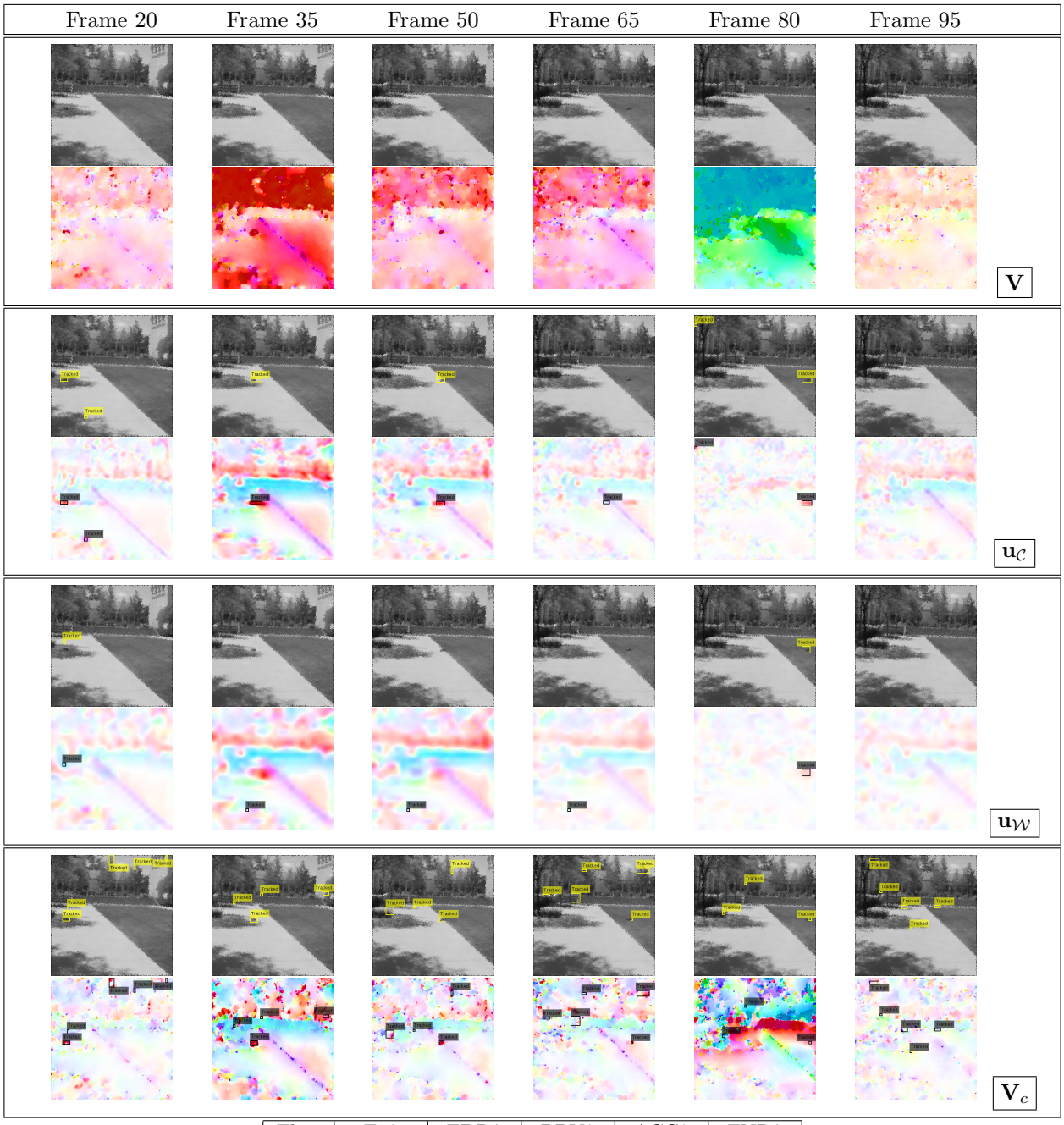


Figure 9: Detection and tracking result for *Courtyard4* at frame 30 with the top presenting the detected mask and the bottom is the magnitude of the magnitude of the flow with the computed threshold shown in red.

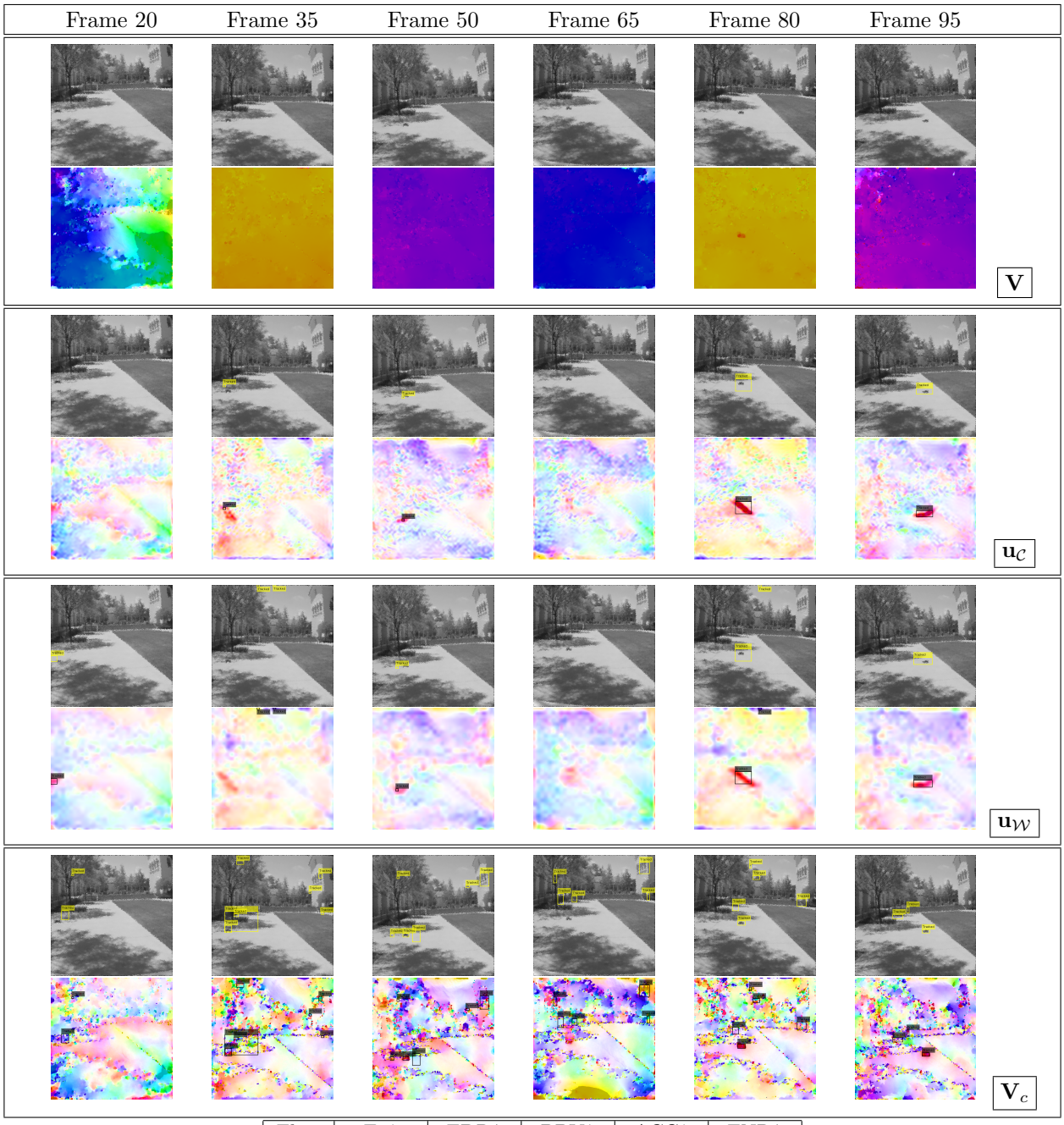
the Kalman Filter requires one of two motion models: constant velocity or constant acceleration. If the motion is assumed to be linear then the constant velocity is the best choice, and if the motion is assumed nonlinear then the constant acceleration is the better option. We assume that in this application the motion from the camera will be coupled with that of a moving object, hence a nonlinear motion model is a preferred choice. Proceeding, any point that has passed the acceptance criterion is assigned to a track, and is annotated on both the sequence frame and its mask. The tracking algorithm then proceeds to the next frame and continues until termination. The results from applying the proposed method are shown in Figures 10-12 where sequences with both simulated atmospheric turbulence and naturally forming atmospheric turbulence are demonstrated.

Each sequence is scored with a performance metric for the curvelet ($\mathbf{u}_{\mathcal{C}}$), wavelet ($\mathbf{u}_{\mathcal{W}}$), and compensated optical flow V_c (*i.e.* without any decomposition) and are shown in a table below each figure. In each table, five different metrics are computed in order to provide a quantitative analysis. Next to each metric we put an $\uparrow$ to denote the best score possible being one and $\downarrow$ being zero. In this work we utilize the following metrics: the F-1 Score (F1$\uparrow$), positive predictive value (PPV$\uparrow$), false discovery rate rate (FDR$\downarrow$), accuracy (ACC$\uparrow$) and false negative rate (FNR$\downarrow$). Detailed information on the computation of these metrics is presented by Fawcett.[27] Due to the amount of generated atmospheric turbulence, the camera motion compensated flow field is unable to discern the moving object from its surroundings. However, once the geometric component of the same field is investigated, the moving object is located and tracked. In Figure 12, due to the lack of atmospheric turbulence that was formed, both algorithms perform comparably. The performance analysis is computed by providing a ground truth structure from the ground truth application mentioned in Section 1.1.



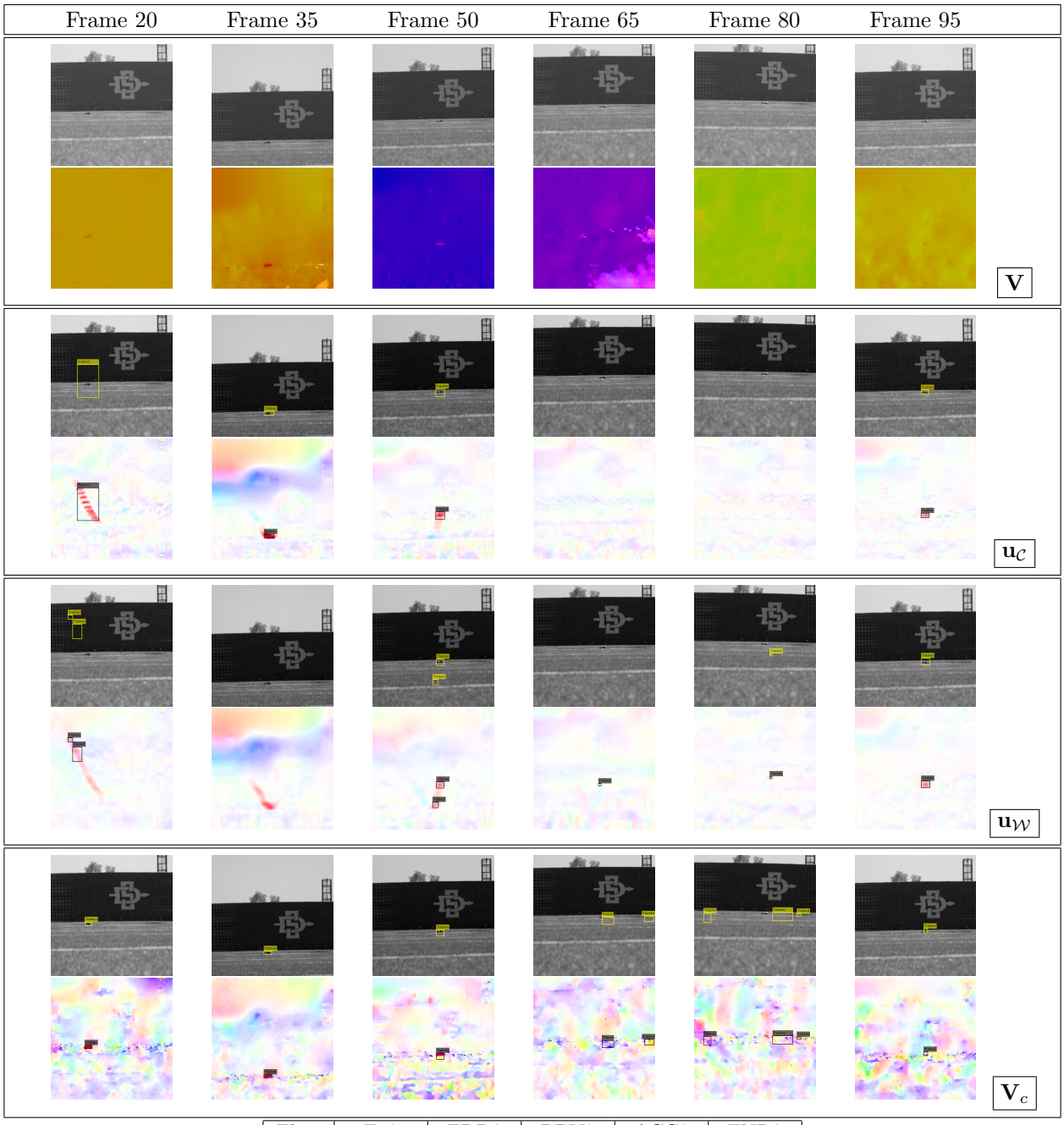
Flow	F1↑	FDR↓	PPV↑	ACC↑	FNR↓
$\mathbf{u}_{\mathcal{C}}$	**0.7351**	**0.1920**	**0.8080**	**0.7188**	0.0893
$\mathbf{u}_{\mathcal{W}}$	0.3601	0.4911	0.5089	0.3571	0.1518
$\mathbf{V}_c$	0.4408	0.6138	0.3862	0.3772	**0.0089**

Figure 10: Results from Courtyard2 which contains simulated atmospheric turbulence. Detection results are shown in image domain as well as a colorized version of the optical flow as to give context to the reader.



Flow	F1↑	FDR↓	PPV↑	ACC↑	FNR↓
$\mathbf{u}_{\mathcal{C}}$	**0.5926**	**0.1414**	**0.8586**	**0.5859**	0.2727
$\mathbf{u}_{\mathcal{W}}$	0.3973	0.2475	0.7525	0.3889	0.3636
$\mathbf{V}_c$	0.4471	0.5047	0.4953	0.3741	**0.1212**

Figure 11: Results from Courtyard4 which contains simulated atmospheric turbulence. Take note of the multiple false positive results in $\mathbf{V}_c$ from the simulated atmospheric turbulence.



Flow	F1↑	FDR↓	PPV↑	ACC↑	FNR↓
$\mathbf{u}_{\mathcal{C}}$	**0.7374**	**0.1818**	**0.8182**	**0.7374**	0.0808
$\mathbf{u}_{\mathcal{W}}$	0.2862	0.5000	0.5000	0.2778	0.2222
$\mathbf{V}_c$	0.4158	0.5404	0.4596	0.3990	**0.0606**

Figure 12: Results from Field1 which contains naturally forming atmospheric turbulence. Take note of the multiple false positive results in $\mathbf{V}_c$ from the naturally forming atmospheric turbulence. Secondarily, take note of the smearing in $\mathbf{u}_{\mathcal{C}}$ and $\mathbf{u}_{\mathcal{W}}$ due to the displacement of the object across the window over few frames due to camera motion.

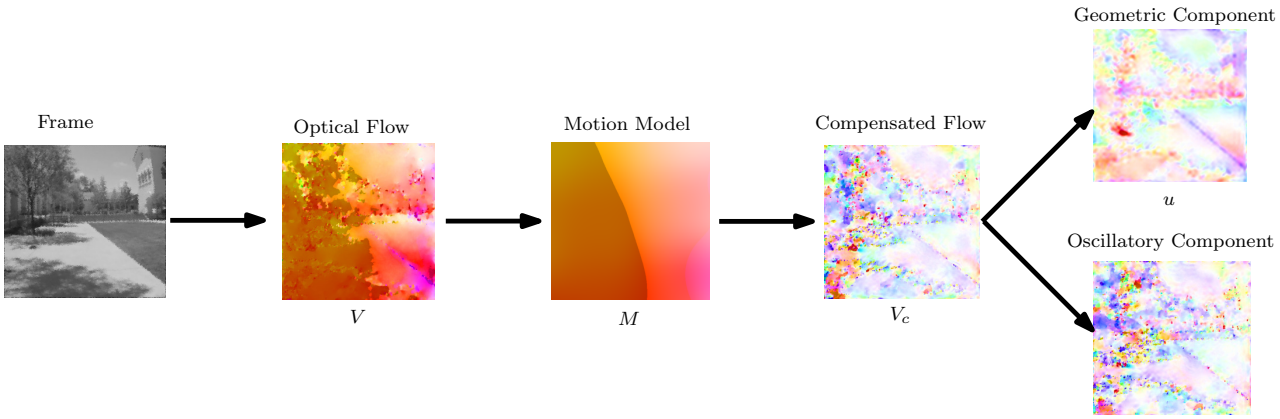


Figure 13: The decomposition process for the simulated turbulence sequence *Courtyard4* at frame 30 when using the empirical camera motion model and curvelet decomposition.

5. CONCLUSION

In this study we have discussed the creation of an image set that contains global camera motion with both simulated or natural atmospheric turbulence as well as the development of an algorithm that detects moving objects from the procured image set. The demonstrated algorithm is able to take an input data sequence, determine its optical flow, and decompose it into its camera motion, geometric, and oscillatory flow fields as summarized in Figure 13. Next, we perform a detection and tracking method on the geometric flow field in order to determine locations within the image set where moving objects exist. The detection of these objects, as well as the preprocessing steps taken to extract the camera motion flow, provide a novel solution to a previously understudied problem. In future work, we will address extensions in four fronts: the inclusion of depth in the motion compensation, the runtime of the cartoon+texture decomposition, the inclusion of camera motion in the Kalman filter prediction step, and finally the confirmation on more datasets.

In the model for global motion flow we have assumed a smooth function to describe the change in $\mathcal{Z}$. In reality, this may not be a fair assumption as a change in depth can be instantaneous. From the camera's perspective, an object that is occluding the field of view *i.e.* a mountain range or a building, may introduce such phenomena. This sharp change will cause a step discontinuity in the optical flow components. In order to handle this step discontinuity, we require a method that segments the flow into regions corresponding to each region

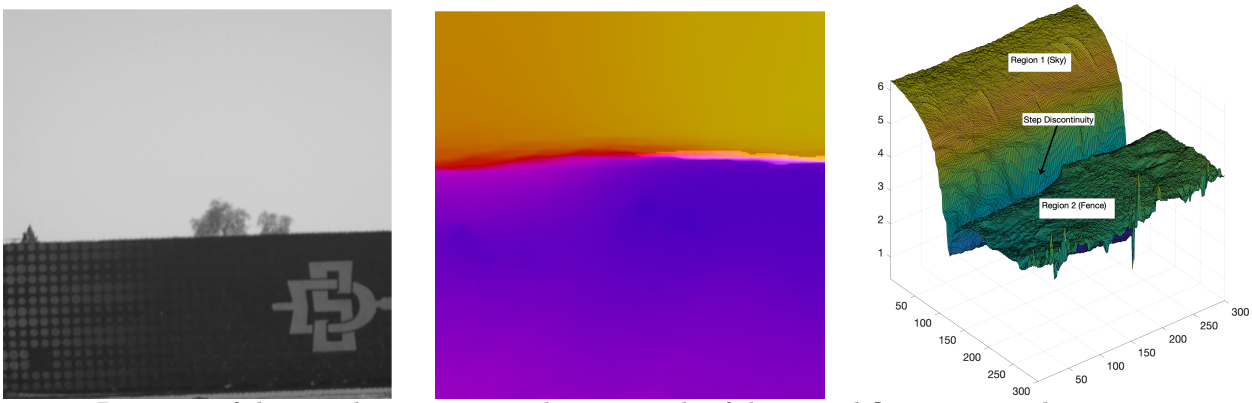


Figure 14: Depiction of the step discontinuity in the magnitude of the optical flow corresponding to an instantaneous change in depth $\mathcal{Z}$. Frame 90 from Field4 (left) is presented with a colorized depiction of the optical flow (center) and the magnitude of the optical flow (right).

of depth. An example of this step discontinuity is shown in Figure 14 where the fence in the sequence causes an instantaneous change in the depth of the image field of view. Once the regions are segmented, the empirical Gaussian smoothing method will provide a motion model for each region.

Recalling Figure 8 we saw the curvelet decomposition did not scale well to larger video sizes. This is a shortcoming that renders real time application impossible. We saw that a naive implementation of the wavelet transform provides acceptable results but failed to capture detailed temporal information. This is certainly due to the bias from scaling the spatial domain equally with the temporal domain. With a sequence with only a few frames, we have rich temporal information over small scales, while a larger video size has no need for such detailed information. Therefore an implementation of a two-dimensional wavelet transform over the spatial information and a one-dimensional transformation over the temporal information is suggested. This method should preserve the wavelet runtime, while providing a fair weighting between the scales for the problem. This approach would be also lend well to coupling sequential data sequences. The current decomposition process operates on three-dimensional data cubes, making coupling another data set difficult to implement. By structuring the decomposition in a way that is separating the spatial and temporal information, it motivates the idea that we may couple data by only taking a new spatial wavelet transform and adding the new temporal information from a new frame.

When using the detection and tracking algorithm, we notice that, even with the inclusion of false positive detections, the algorithm is able to detect and track the moving object accurately. During the implementation of the Kalman Filter, we provide simply a constant acceleration motion model, without the input of the model for camera motion. Without knowledge of camera motion, the Kalman filter is not working as efficiently as possible as its predictions are impacted by the camera motion. As we have already built a motion model, we have a prediction for the location of each pixel in the next frame. If we were to couple this knowledge with the Kalman filter prediction step, a more accurate tracking algorithm could be created.

Finally, this work only contained a limited data set, some of which did not contain heavy enough atmospheric turbulence to offset the speed at which the object was moving. As we detect the object by adaptive thresholding of the speed of the flow, we expect that a fast moving object will be detected in most situations. A dataset that contains a slow moving object with a moving camera would be a great demonstration and further confirmation of this promising algorithm.

6. ACKNOWLEDGMENT

This work was supported by the Air Force Office of Scientific Research under the grant number FA9550-15-1-0065.

APPENDIX A. CAMERA MOTION MODEL DERIVATION

We previously observed that the flow induced by camera motion is a smooth function by appearance (see Figure 3). One model for the optical flow induced by global camera motion is derived from the pinhole camera model (16) and has been previously presented by Thompson and Pong[28] as well as Trucco and Verri.[21] From Trucco and Verri, the pinhole camera model is given by (16)

$$\mathbf{p} = \mathfrak{f}\frac{\mathcal{P}}{\mathcal{Z}} \tag{16}$$

where $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ in the usual 3D camera reference basis. The projection center is chosen to be the origin of that reference basis and $\mathfrak{f}$ denotes the focal length. This model provides a relation between points in the image scene $\mathcal{P}$ and their projections in the image plane $\mathbf{p} = [x, y, \mathfrak{f}]^\top$. As the image plane is at a constant location along the optical axis, the third coordinate $\mathfrak{f}$ is dropped from the notation to have the point in the image plane $\mathbf{p} = [x, y]^\top$. A depiction of this projection, from Vismara[29] is shown in Figure 15. The relative motion between $\mathcal{P}$ and the camera is described as

$$\mathcal{V} = -\mathbf{T} - \omega \times \mathcal{P}, \tag{17}$$

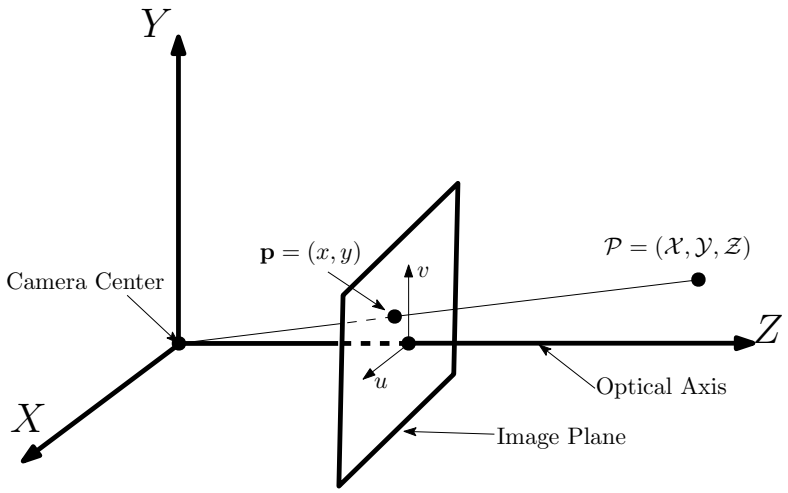


Figure 15: Illustration of the pinhole camera model having the focal plane in front of the camera center. The image point $\mathbf{p}$ is located at $\mathfrak{f}$ on the optical axis Z.

where $\mathbf{T} = (T_x, T_y, T_z)$ is the translation component of motion and $\omega = (\omega_x, \omega_y, \omega_z)$ the angular component which are constant at a given frame for all arbitrary points $\mathcal{P}$ as we have rigid body camera motion. We denote $\mathbf{P}, \mathbf{Y}, \mathbf{R}$ the optical flows induced by pitch (*i.e.* by ω_x), yaw (*i.e.* by ω_y) and roll (*i.e.* by ω_z). Then (17) expressed by its coordinates becomes,

$$\begin{aligned}
\mathcal{V}_x &= -T_x - \omega_y \mathcal{Z} + \omega_z \mathcal{Y} \\
\mathcal{V}_y &= -T_y - \omega_z \mathcal{X} + \omega_x \mathcal{Z} \\
\mathcal{V}_z &= -T_z - \omega_x \mathcal{Y} + \omega_y \mathcal{X}.
\end{aligned} \tag{18}$$

To obtain the relation between the velocity of $\mathcal{P}$ in real-world coordinates and the corresponding velocity of $\mathbf{p}$ on the image plane, we take the time derivative of both sides of equation (16) to obtain through the quotient rule our predicted optical flow,

$$\mathbf{V} = \mathfrak{f} \frac{\mathcal{Z}\mathcal{V} - \mathcal{V}_z \mathcal{P}}{\mathcal{Z}^2}, \tag{19}$$

where by substituting (17) and (18) into (19) we get

$$\begin{aligned}
V_x &= \mathfrak{f} \frac{\mathcal{Z}\mathcal{V}_x - \mathcal{V}_z \mathcal{X}}{\mathcal{Z}^2} \\
V_y &= \mathfrak{f} \frac{\mathcal{Z}\mathcal{V}_y - \mathcal{V}_z Y}{\mathcal{Z}^2}.
\end{aligned} \tag{20}$$

Converting from real world coordinates to 2D image plane coordinates, we get

$$\begin{aligned}
V_x &= \mathfrak{f} \frac{\mathcal{Z}\mathcal{V}_x - V_z \mathcal{Z} \frac{x}{\mathfrak{f}}}{\mathcal{Z}^2} = \mathfrak{f} \frac{\mathcal{V}_x - \mathcal{V}_z \frac{x}{\mathfrak{f}}}{\mathcal{Z}} \\
V_y &= \mathfrak{f} \frac{\mathcal{Z}\mathcal{V}_y - V_z \mathcal{Z} \frac{y}{\mathfrak{f}}}{\mathcal{Z}^2} = \mathfrak{f} \frac{\mathcal{V}_y - \mathcal{V}_z \frac{y}{\mathfrak{f}}}{\mathcal{Z}}
\end{aligned} \tag{21}$$

and expanding $\mathcal{V}_x$, $\mathcal{V}_y$, $\mathcal{V}_z$ into 2D pixel coordinates, we get

$$\begin{aligned}
\mathcal{V}_x &= -T_x - \omega_y \mathcal{Z} + \omega_z \frac{y}{\mathfrak{f}} \\
\mathcal{V}_y &= -T_y - \omega_z \frac{x}{\mathfrak{f}} + \omega_x \mathcal{Z} \\
\mathcal{V}_z &= -T_z - \omega_x \frac{y}{\mathfrak{f}} + \omega_y \frac{x}{\mathfrak{f}},
\end{aligned} \tag{22}$$

then by combining (21) and (22) we finally obtain,

$$V_x = \frac{T_z x - T_x \mathfrak{f}}{\mathcal{Z}} + \omega_x \frac{xy}{\mathfrak{f}} - \omega_y \left(\mathfrak{f} + \frac{x^2}{\mathfrak{f}} \right) + \omega_z y$$
$$V_y = \frac{T_z y - T_y \mathfrak{f}}{\mathcal{Z}} + \omega_x \left(\mathfrak{f} + \frac{y^2}{\mathfrak{f}} \right) - \omega_y \frac{xy}{\mathfrak{f}} - \omega_z x. \tag{23}$$

We observe that the velocity field is the sum of two components, one containing information about camera translation and the other containing information about rotation. Hence we denote the translation components to be

$$V_x^T = \frac{T_z x - T_x \mathfrak{f}}{\mathcal{Z}}$$
$$V_y^T = \frac{T_z y - T_y \mathfrak{f}}{\mathcal{Z}}, \tag{24}$$

and the rotation components to be,

$$V_x^\omega = \omega_x \frac{xy}{\mathfrak{f}} - \omega_y \left(\mathfrak{f} + \frac{x^2}{\mathfrak{f}} \right) + \omega_z y$$
$$V_y^\omega = \omega_x \left(\mathfrak{f} + \frac{y^2}{\mathfrak{f}} \right) - \omega_y \frac{xy}{\mathfrak{f}} - \omega_z x. \tag{25}$$

We notice that in each velocity component, information on depth $\mathcal{Z}$ and rotation ω are decoupled. This shows that the part of the velocity field that depends on angular velocity does not carry information on depth

REFERENCES

[1] Gilles, J., Alvarez, F., Ferrante, N., Fortman, M., Tahir, L., Tarter, A., and von Seeger, A., "Detection of moving objects through turbulent media. decomposition of oscillatory vs non-oscillatory spatio-temporal vector fields," *Image and Vision Computing* **73**, 40–55 (2018).

[2] Shaikh, S. H., Saeed, K., and Chaki, N., "Moving object detection using background subtraction," in [*Moving Object Detection Using Background Subtraction*], 15–23, Springer (2014).

[3] Yi, Z. and Liangzhong, F., "Moving object detection based on running average background and temporal difference," in [*Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on*], 270–272, IEEE (2010).

[4] Chauhan, A. K. and Krishan, P., "Moving object tracking using gaussian mixture model and optical flow," *International Journal of Advanced Research in Computer Science and Software Engineering* **3**(4) (2013).

[5] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning," *nature* **521**(7553), 436 (2015).

[6] Patel, H. A. and Thakore, D. G., "Moving object tracking using Kalman filter," *International Journal of Computer Science and Mobile Computing* **2**(4), 326–332 (2013).

[7] Gordon, N., Ristic, B., and Arulampalam, S., "Beyond the Kalman filter: Particle filters for tracking applications," *Artech House, London* **830**, 5 (2004).

[8] Briechle, K. and Hanebeck, U. D., "Template matching using fast normalized cross correlation," in [*Optical Pattern Recognition XII*], **4387**, 95–103, International Society for Optics and Photonics (2001).

[9] Comaniciu, D., Ramesh, V., and Meer, P., "Real-time tracking of non-rigid objects using mean shift," in [*Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*], **2**, 142–149, IEEE (2000).

[10] Avidan, S., "Support vector tracking," in [*Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*], **1**, I–I, IEEE (2001).

[11] Zhou, Y. and Tao, H., "A background layer model for object tracking through occlusion," in [*Proceedings Ninth IEEE International Conference on Computer Vision*], 1079–1085, IEEE (2003).

[12] Rosenhahn, B., Brox, T., Cremers, D., and Seidel, H.-P., "A comparison of shape matching methods for contour based pose estimation," in [*International Workshop on Combinatorial Image Analysis*], 263–276, Springer (2006).

[13] Balaji, S. and Karthikeyan, S., "A survey on moving object tracking using image processing," in [*2017 11th international conference on intelligent systems and control (ISCO)*], 469–474, IEEE (2017).

[14] Yazdi, M. and Bouwmans, T., "New trends on moving object detection in video images captured by a moving camera: A survey," *Computer Science Review* **28**, 157–177 (2018).

[15] Giancoli, D. C., [*Physics for scientists & engineers with modern physics*], vol. 2, Pearson Education (2008).

[16] Hyde, M. W., Schmidt, J. D., Havrilla, M. J., and Cain, S. C., "Determining the complex index of refraction of an unknown object using turbulence-degraded polarimetric imagery," Tech. Rep. 12 (2010).

[17] Gilles, J. and Ferrante, N. B., "Open turbulent image set (OTIS)," *Pattern Recognition Letters* **86**, 38–41 (2017).

[18] Tahtali, M., Fraser, D., and Lambert, A., "Restoration of non-uniformly warped images using a typical frame as prototype," in [*TENCON 2005 2005 IEEE Region 10*], 1–6, IEEE (2005).

[19] Horn, B. K. and Schunck, B. G., "Determining optical flow," *Artificial intelligence* **17**(1-3), 185–203 (1981).

[20] Zach, C., Pock, T., and Bischof, H., "A duality based approach for realtime TV-L1 optical flow," in [*Joint Pattern Recognition Symposium*], 214–223, Springer (2007).

[21] Trucco, E. and Verri, A., [*Introductory techniques for 3-D computer vision*], vol. 201, Prentice Hall Englewood Cliffs (1998).

[22] Candès, E. J., Demanet, L., Donoho, D. L., and Ying, L., "Fast discrete curvelet transforms," *Multiscale Modeling and Simulation* **5**(3), 861–899 (2005).

[23] Kohavi, R. et al., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in [*Ijcai*], **14**(2), 1137–1145, Montreal, Canada (1995).

[24] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering* **82**(1), 35–45 (1960).

[25] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics* **5**(1), 32–38 (1957).

[26] Miller, M. L., Stone, H. S., and Cox, I. J., "Optimizing murty's ranked assignment method," *IEEE Transactions on Aerospace and Electronic Systems* **33**(3), 851–862 (1997).

[27] Fawcett, T., "An introduction to roc analysis," *Pattern recognition letters* **27**(8), 861–874 (2006).

[28] Thompson, W. B. and Pong, T.-C., "Detecting moving objects," *International journal of computer vision* **4**(1), 39–57 (1990).

[29] VISMARA, C., "Monitoring human state in a robotic assistive platform: data acquisition and person detection systems," (2015).