2-2014

Optical Flow Background Estimation for Real-time Pan/tilt Camera Object Tracking

Daniel D. Doyle

Alan L. Jennings
Air Force Institute of Technology

Jonathan T. Black
Air Force Institute of Technology

Measurement 48 (2014) 195–207

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

Optical flow background estimation for real-time pan/tilt camera object tracking☆☆

Daniel D. Doyle *, Alan L. Jennings, Jonathan T. Black

Department of Aeronautical and Astronautical Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, United States

ARTICLE INFO

Article history:
Received 24 May 2013
Received in revised form 13 August 2013
Accepted 10 October 2013
Available online 23 October 2013

Keywords:
Pan/tilt camera
Optical flow
Background subtraction
Object tracking

ABSTRACT

As Computer Vision (CV) techniques develop, pan/tilt camera systems are able to enhance data capture capabilities over static camera systems. In order for these systems to be effective for metrology purposes, they will need to respond to the test article in real-time with a minimum of additional uncertainty. A methodology is presented here for obtaining high-resolution, high frame-rate images, of objects traveling at speeds $\geqslant$1.2 m/s at 1 m from the camera by tracking the moving texture of an object. Strong corners are determined and used as flow points using implementations on a graphic processing unit (GPU), resulting in significant speed-up over central processing units (CPU). Based on directed pan/tilt motion, a pixel-to-pixel relationship is used to estimate whether optical flow points fit background motion, dynamic motion or noise. To smooth variation, a two-dimensional position and velocity vector is used with a Kalman filter to predict the next required position of the camera so the object stays centered in the image. High resolution images can be stored by a parallel process resulting in a high frame rate procession of images for post-processing. The results provide real-time tracking on a portable system using a pan/tilt unit for generic moving targets where no training is required and camera motion is observed from high accuracy encoders opposed to image correlation.

1. Introduction

1.1. Motivation

The Unmanned Aircraft System (UAS) flight plan describes Nano/Micro UAS as "aircraft capable of conducting a variety of indoor and outdoor reconnaissance sensing missions" [1]. Developing aircraft capable of these activities requires an in-depth understanding of biological counterparts and measurement systems capable of capturing in-flight dynamics. In order to gain this type of understanding for biomimetics advancement, measurement systems require the ability to measure the motion of flying organisms in an unconstrained, real-time manner.

The Air Force Institute of Technology and the Air Force Research Laboratory have worked on advancing flapping wing mechanisms through flight control research, wing design and analysis of dynamics [2–4]. There are many other facilities devoted to developing and understanding the mechanics of flapping wing flight. A sample includes: the Harvard Microrobotics Lab [5], the Korean Advanced Institute of Science and Technology (KAIST) Smart Systems and Structures Lab [6], and Georgia Institute of Technology's Intelligent Control Systems Laboratory [7]. Due to the small scale and speed involved with these vehicles, novel systems have been needed to measure forces and motion without disrupting flight.

☆☆ The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

* Corresponding author. Tel.: +1 937 656 8743.
E-mail addresses: daniel.doyle@us.af.mil (D.D. Doyle), alan.jennings@afit.edu (A.L. Jennings), jonathan.black@afit.edu (J.T. Black).

0263-2241
http://dx.doi.org/10.1016/j.measurement.2013.10.025

Contact measurement methods pose many challenges when working with UASs. These methods consist of, but are not limited to, using strain gages, paint, retro-reflective markers, etc. which typically alter vehicle performance either by weight or other restrictions to an already lightweight device (typically ~15 gm). The alternative approach is using non-contact measurement methods for observing the UAS's behavior without disrupting it. Current non-contact displacement measurement methods include laser vibrometry, laser range finders, capacitance measurement, interferometry and photo/videogrammetry. Laser vibrometry is the preferred method for measuring non-contact vibrations [8]. However, only a single point is measured at a time, and alignment must be maintained through the test. Videogrammetry uses multiple, synchronized images or stereo pairs and typically retro-reflective markers or projected targets for determining surface shapes and can capture shape and motion from a single test [4]. Projected targets are not practical for use with moving targets, and adding retro-reflective markers negates non-contact measurement. Using a camera allows for more freedom of movement. However, the subject must stay in the field-of-views (FOVs) with little to no occlusion. Incorporating a pan/tilt camera with real-time tracking will allow for the camera to move so that the object remains in the FOV. This will expand the effective capture volume without a sacrifice in resolution of the object, for better flight data over a larger set of flight profiles. The ideal system would have a fast response (high frame rate and low latency allows for faster vehicles), high resolution (for better discrimination of the vehicle) and work with general dynamic objects (not require selection or training).

1.2. Related work

Surveillance and robotics have driven development of computer vision (CV) tools. The CV tools related to this work involve camera motion compensation, object detection, and object tracking. Features discriminate a point from others, such as differences in color or intensity gradients. Different methods can be employed to detect specific features or objects in the scene; such as point detection, background modeling, image segmentation or classifiers based on supervised learning. Tracking can be done by matching points, or objects can be tracked as a whole by employing silhouettes or kernels. See Yilmaz et al. [9] for a detailed survey on object tracking.

The system presented here pulls existing methods in a novel combination for general purpose. It uses known camera motion with exact pixel displacement, corner detection and optical flow tracking. Image processing is performed on the graphics processing unit (GPU) for faster speed. The resulting system is able to track a single moving object using a single camera.

Camera motion can either be undesired, requiring image stabilization [10], or desired, such as for tracking or building a background mosaic [11]. The principle involved is the same: finding the transform from an image to a reference image [12]. Transforms are typically found between frames by fitting or matching point correspondences [13], phase correlation [14] or block matching [15]. An alternate technique is to use a projective texture as a Ref. [12]. The use of the known camera motion is surprisingly not used more often. Though the exact model is developed [16], often simplified models are used, such as the small angle approximation [17]. Algorithms that do not require known camera motion have the advantage that they can be used with hand-controlled cameras or other moving mounts. However, estimating motion comes at the cost of increased error, increased processing time, the object appearing smaller in the image (because the background must be viewed) and requiring texture, or the lack of texture, on the background and foreground. This work uses the exact model based on encoder positions for high accuracy predictions, even with large displacements (~1/7th of the image).

An alternative to compensating for tracking the camera's motion is to use one or more static cameras to direct the moving camera. This has the advantage that moving objects are simpler to detect with a static camera. Applications include controlling a remote vehicle [18,19] or following sporting events [20]. For this alternative to be effective, the FOV must cover the range of the moving camera, increasing setup time and calibration. However, stationary, smart cameras can be used to self-calibrate with fast setup and reasonable accuracy [21]. The work presented here uses a single pan/tilt camera to track a moving object.

As processing power has developed, new and existing tools have been applied in real-time, and have been used for both tracking an object within a static camera's view or tracking an object with a moving camera.

Heuristics can be used to design detectors for increased accuracy or speed. General purpose feature descriptors can be used with training data, tuning or selection for a wide variety of objects. Some specific applications using a pan/tilt camera include tracking the color on a ball [22,23], human positioning/tracking [24–26], faces [23,26,27], and moving vehicles/ships [13,27–29]. Various methods are used in these applications for detecting objects such as mean shift and its variant continuously adaptive mean shift (CAMShift) [22,25,26,28,29], background modeling using a Gaussian mixture model (GMM) [25,29], skin color segmentation [23], training/learning an object's appearance using principle component analysis (PCA) and sum of squared difference (SSD) [27], and template-based detection using speeded up robust features (SURF) [13]. The general purpose methods often require selection, a pre-defined capture area (i.e., alert zone), training, and/or tuning. The cost of using application specific algorithms include decreased system versatility, increased setup/calibration time, and limitations on feature changes such as illumination, object pose, shape, and size.

Features can also be non-specific, such as motion-based features, which are useful in general purpose applications. General purpose (motion-based) tracking allows for more flexibility in handling multi-purpose problems and allows tracking of an object of any size or shape [17]. This is well-suited for a metrology system capable of tracking a multitude of objects without having the concern of training, or changes in illumination or appearance associated with application specific methods. Short duration/sudden events are easily accommodated. Operator delay is

Table 1
Related work summary of pan/tilt system resolution and operating frame rate.

Ref.	Resolution (width × height)	Frame rate (pan/tilt delay)	Training	Occlusion handling
[11]	320 × 240	11 (unknown)	No	No
[12]	Not available	22 (25 ms)	Yes	No
[13]	800 × 600	~30 (unknown)	Yes	Partial
[14]	Not available	8 (unknown)	No	No
[15]	320 × 240	5 (unknown)	Yes	No
[17]	Not available	Unknown (unknown)	No	Partial
[30]	320 × 240	Not available	No	Partial
[24]	640 × 480	30 (unknown)	No	No
[27]	1024 × 768	20–25 (unknown)	Yes	Full
[28]	320 × 240	25 (unknown)	No	Partial
[29]	768 × 576	25 (unknown)	No	Partial
[22]	320 × 240	10 (unknown)	No	Partial
[26]	352 × 255	20–25 (100 ms)	No	Partial
[23]	Not available	Unknown (unknown)	No	Partial

removed since object selection and reselection are not required.

It is sought to compare related work using standard system characteristics such as image resolution and frame rate. In addition, many CV algorithms require training and design for occlusion handling. Therefore, Table 1 provides a comparison of related work by resolution, frame rate, training and occlusion handling of related PTZ systems.

The method presented in this paper uses Kanatani's derivation of pixel position between images taken from different positions of rotation about the lens center [16]. Murray et al. [17] similarly use Kanatani's relationship of knowing the pan/tilt motion in supporting a motion tracking system to compensate for the background through image subtraction, edge detection, morphological filtering and thresholding. Their technique employs a motion-energy method vs. the optic flow tracking employed in this paper.

1.3. Overview of this approach

This work is novel in that it is suited for all textured moving objects (or general purpose) with the application intent of tracking an indoor UAS. Due to the speed of indoor UASs, cameras must pan and tilt to track the motion, and CV algorithms must run at least at 30 fps. If the camera captured the whole scene, then the resolution would be unusable. Pan/tilt cameras have the advantage that the view is not fixed, but every motion of the camera causes apparent motion of points which are actually static. This apparent motion, termed *background motion*, can be calculated by Kanatani's relationship [16]. *Background motion* must be known to determine the actual motion of moving points, termed *derived motion*.

The major contributions of this paper are the analysis and experimental results of Kanatani's relationship on optical flow tracking, and its implementation with GPU-based optical flow for real-time, general purpose tracking. The analytical results show the induced error with respect to geometric approximations. The experimental results use the static feature points with measured camera motion to determine the error introduced by optical flow techniques. Known motion and inertial measurement have been used to compensate for motion [10,17], but using commanded motion with optical flow for exact *derived motion* and background classification has not been seen by the authors to date. GPU-based feature detection and tracking is an advancement beyond the methods presented in Section 1.2.

The goal of this work is to develop a system to track a single moving object (specifically a UAS) with a pan/tilt camera. Processing uses a GPU-based, good-features-to-track detector[1] (e.g. corners) and a GPU-based implementation of the pyramidal Lucas-Kanade algorithm[2] [31]. Good features are detected in an image and placed as optical flow points. The *background motion* is calculated for each of these points in order to extract the *derived motion*. Optical flow points with a *derived motion* less than a threshold are considered background. Moving points can then be used for CV-based tracking.

Some possible applications include: tracking and testing of flexible vehicles and structures without the use of tracking aids, tracking or monitoring vehicles, to include satellites for intercepting space debris, incoming/departing aircraft from a flight tower, and obtaining high-quality images of UASs, satellites, debris, intruders, aircraft, etc.

The system components are described in Section 2. The theory and extended approximations for application dependent systems is provided in Section 3. The algorithm development is provided in Section 4. The experiments and results are provided in Section 5. The conclusions and future work are described in Section 6.

2. System components

2.1. Equipment used

The equipment used for this paper include a Dell M6600 Precision laptop with Intel® Core™ i7-2820QM central processing unit (CPU) @2.3 GHz with a NVIDIA Quadro 4000M graphics card and 2.0 GB dedicated graphics memory, two PTU-D46-17 pan/tilt units, and two PointGrey Research, Inc. USB 3.0 Flea-3 FL3-U3-13S2C-CS (1328 × 1048) CMOS color cameras, two Fujinon Lens-30F2-VC0CS and one USB to four channel RS-232 communication module: FTDI

[1] OpenCV 2.4.2 Library – GoodFeaturesToTrackDetector_GPU.
[2] OpenCV 2.4.2 Library – PyrLKOpticalFlow (GPU).

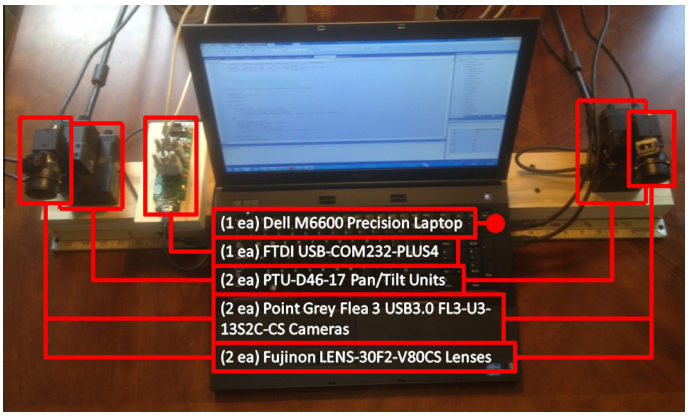

Fig. 1. Equipment used.

USB-COM232-PLUS4 (see Fig. 1). The requested image size for the experiments is 656 × 524 with a requested shutter speed of 1.02 ms. The focal length is 1076 pixels.

2.2. Background and system under test (SUT)

A portion of the background used for the initial performance testing is provided in Fig. 2.

The System Under Test (SUT) is a hanging UAS (see Fig. 3) suspended as a pendulum. The large, red circle indicates the calculated pixel centroid, $(x,y)_{pix}$, and the green rectangles represent the detected features, $\mathbf{g}_i$. The length of the pendulum is approximately 1.2 m (47 in.), and so the dynamics of the system are known using

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where T is the period, g is gravity, and L is the pendulum length. A period of 2.2 s was determined experimentally over ten cycles which agrees with the theoretical period. Given the period, the average velocity of the SUT is calculated using twice the length of the arc divided by 2.2 s.

3. Theoretical development

3.1. Background optical flow estimation

The motion of the pan/tilt unit is modeled here for determining where background optical flow points should theoretically move based on pan/tilt camera movement. It is assumed that the rotation is about the camera center. A change in the camera coordinate system due to pan and tilt motions leads to *background motion* in an image. This *background motion*, which is the change in the position of stationary, global points in the camera coordinate system, is estimated in terms of pixel movement in an image.

Eq. (2) provides the rotation matrix for pan/tilt motions where the z-axis represents the principal axis, the x-axis is positive, right and the y-axis is positive, down. Pre-multiplying a set of points in global coordinates by

Fig. 2. Background used for performance testing.

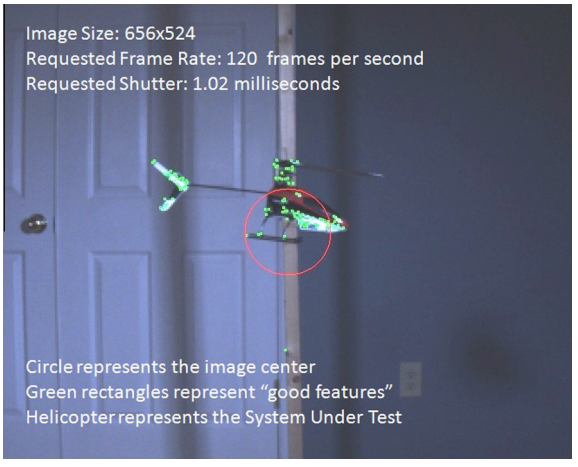

Fig. 3. System Under Test (SUT) is a hanging helicopter that acts as a pendulum for tracking; green squares represent the initial placement of the optical flow points from image, I_{i-1} and the large, red circle represents the calculated pixel centroid, $(x,y)_{pix}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the rotation matrix, R, then gives the new locations of the points in (3) as seen by the camera.

$$\mathbf{R}(\psi,\theta)=\begin{bmatrix}\cos(\psi) & 0 & -\sin(\psi)\\ \sin(\psi)\sin(\theta) & \cos(\theta) & \cos(\psi)\sin(\theta)\\ \sin(\psi)\cos(\theta) & -\sin(\theta) & \cos(\psi)\cos(\theta)\end{bmatrix} \tag{2}$$

$$\mathbf{X}_{i+1}=\mathbf{R}(\psi_i,\theta_i)\mathbf{X}_i \tag{3}$$

where $\mathbf{X}_i$ is the 3D location of the point with respect to the camera coordinate system, i is the frame number in a sequence of images, ψ_i and θ_i are the change in pan and tilt angles from the ith to the $i+1$th coordinate system, respectively. Note that Eq. (3) is use to find the change from the previous camera coordinate system, not the global coordinate system; so only the change in angle is needed, not the absolute angles.

Using the focal length of the camera to provide proportional points on the image plane with respect to three-dimensional coordinates provides the ability to determine optical flow motion of static points given inputs of image coordinates (x,y) with respect to the principal axis, delta pan angle, ψ_i, and delta tilt angle, θ_i. The pinhole camera model

$$(X,Y,Z)^\top \mapsto (f\,X/Z, f\,Y/Z)^\top = (x,y)^\top \tag{4}$$

provides the means for determining the image plane coordinates of a set of points from 3D points.

3.1.1. Closed-form solution

The resulting Eqs. (5) and (6) provide the estimated background location of a point $(\hat{x},\hat{y})_{i+1}$ on the image plane given an input of image coordinates $(x,y)_i$, focal length, f, and delta pan and tilt angles, ψ_i and θ_i. This relationship was first developed by Kanatani [16].

$$\hat{x}_{i+1}=f\left(\frac{x_i-f\tan(\psi_i)}{x_i\tan(\psi_i)\cos(\theta_i)-y_i\frac{\sin(\theta_i)}{\cos(\psi_i)}+f\cos(\theta_i)}\right), \tag{5}$$

$$\hat{y}_{i+1}=f\left(\frac{x_i\sin(\psi_i)\tan(\theta_i)+y_i-f\cos(\psi_i)\tan(\theta_i)}{x_i\sin(\psi_i)-y_i\tan(\theta_i)+f\cos(\psi_i)}\right). \tag{6}$$

3.1.2. Small-angle/first-order approximation

Small movements are made and captured from one frame to the next in the range of 0–3.1° for the experiments given in Section 5. Analysis of using approximations is sought to evaluate the viability of making such assumptions. The small-angle approximation simplifies the closed-form solution from having to use basic trigonometric functions by neglecting any second or higher order terms. The following substitutions are made: $\sin(\Theta)\approx\Theta$, $\cos(\Theta)\approx 1$, and $\tan(\Theta)\approx\Theta$ where Θ is an arbitrary angle in radians. The small-angle/first-order approximation is as follows:

$$\hat{x}_{i+1}=f\left(\frac{x_i-f\psi_i}{x_i\psi_i-y_i\theta_i+f}\right), \tag{7}$$

$$\hat{y}_{i+1}=f\left(\frac{y_i-f\theta_i}{x_i\psi_i-y_i\theta_i+f}\right). \tag{8}$$

3.1.3. Simplified, linear approximation

The small-angle/first-order approximation can be further simplified by assuming f is the dominant term in the denominator of (7) and (8). This gives a simplified, linear approximation of:

$$\hat{x}_{i+1}=x_i-f\psi_i, \tag{9}$$

$$\hat{y}_{i+1}=y_i-f\theta_i. \tag{10}$$

3.1.4. Summary of proposed models

The models above are proposed in hopes that computational efficiency may be found and utilized according to application specific needs. Applications requiring faster processing times, such as embedded microcontrollers, may determine that one of the above approximations is more appropriate.

3.2. Approximation accuracy

Sensor bias may be a problem in any sensor-based application and must be calibrated for application accuracy. Theoretical accuracy of the small-angle/first-order approximation and the simplified, linear approximation compared to the closed-form solution provides insight into application-specific needs. Figs. 4 and 5 show the mean and max differences of the small-angle and linear approximations with respect to the closed-form solution. Note that the rotational matrix, R, in Eq. (2) applies pan and then tilt explaining why axis error values are different. For the mean of a set of points distributed uniformly, the small-angle approximation provides accuracy to within a pixel for $\psi\leqslant 8°$, $\theta=0°$. The linear approximation provides accuracy of a pixel for $\psi\leqslant 2.3°$, $\theta=0°$. Varying the tilt angle for these approximations reduces the theoretical accuracy according to Fig. 4. Asymmetric error is seen in Figs. 4 and 5 primarily due to the closed form solution's application of pan and then tilt operations in the rotation matrix.

Throwing out the assumption of uniform point placement, the worst case error due to the models for a given ψ and θ is shown in Fig. 5 for the small angle approximation and the linear approximation. The maximum approximation pixel error is greater with tilt angle, θ, for the small angle/first-order approximation; however, the error is at least four times smaller than the linear approximation (e.g., $\psi=4°$, $\theta=2°$, the small-angle give 1/2 pixel and the linear gives 7 pixels). Utility of the simplified, linear approximation leans towards applications requiring less than 2.3° of pan/tilt between images.

4. Algorithm development

4.1. Point classification

In the process of detecting "good features", $\mathbf{g}_i$ for point classification, a boundary of 50 pixels from the image edge is omitted to effectively make a 656×524 image, a 556×424 image for analysis. This facilitates better overlap to match "good features" from the previous image, I_{i-1}, to the current image, I_i. Fig. 6 shows three scenarios used to

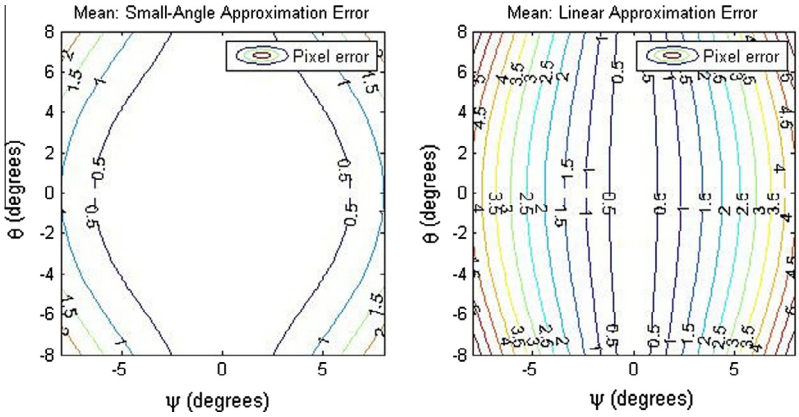

Fig. 4. The mean error of the small-angle approximation (left) and linear approximation (right).

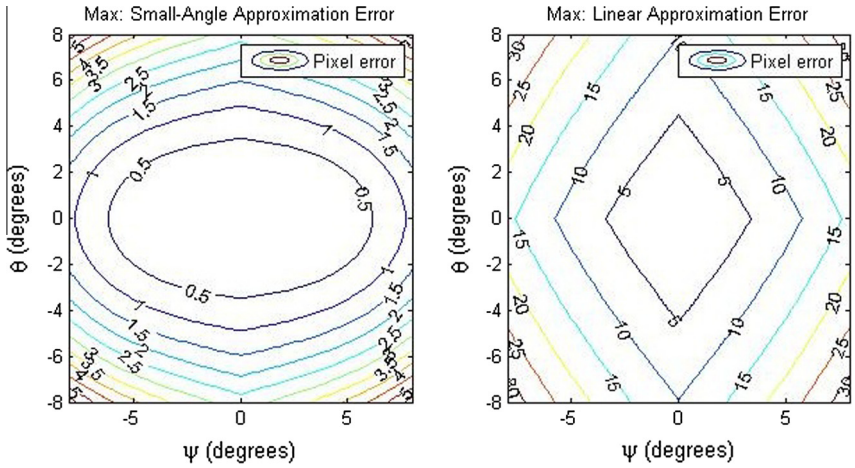

Fig. 5. Closed-form solution versus the max error of the small-angle approximation (left) and linear approximation (right).

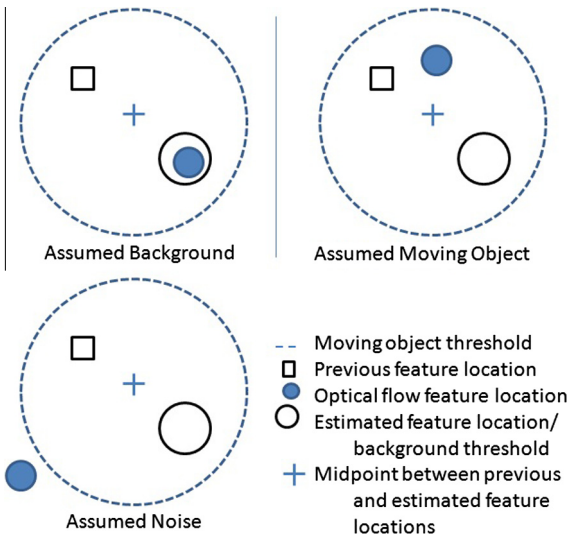

Fig. 6. Interpreting optical flow points in a current frame as background, moving, or noise.

classify optical flow points in a given frame as background, moving, or noise. Given a pan/tilt movement, the midpoint between the previous and estimated feature locations is used to point at a greater distance than the moving object threshold for noise determination. An optical flow point, $\mathbf{g}_i$, lying within the estimated feature location/background threshold, b_{bkgd}, is considered background, otherwise a point within the moving object threshold, d_{moving}, is considered a dynamic point.

The background threshold, b_{bkgd}, is the first parameter used for determining whether $\mathbf{g}_i$ is background or moving using the estimated background location, $\hat{\mathbf{g}}_i$ to give the Euclidean distance in $\mathbf{b}_i$. The second threshold is the moving object threshold which is a circle about the midpoint of "good feature", $\mathbf{g}_i$, and an estimated background location, $\hat{\mathbf{g}}_i$ given by the Euclidean distance in $\mathbf{d}_i$. Therefore, moving or dynamic points in $\mathbf{g}_i$, are classified using the criteria

$$\begin{aligned} &\mathbf{b}_i = \|\mathbf{g}_i - \hat{\mathbf{g}}_i\|, \\ &b_i(n) > b_{bkgd} \quad \forall n \in \{\mathbf{g}_i\} \end{aligned} \tag{11}$$

$$\mathbf{d}_i = \left\| \mathbf{g}_i - \frac{1}{2}(\mathbf{g}_{i-1} + \hat{\mathbf{g}}_i) \right\|, \quad d_i(n) < d_{moving} \quad \forall n \in \{\mathbf{g}_i\} \tag{12}$$

where n is the index of the points in the set of matched features, b_{bkgd} is a given threshold for background determination, and d_{moving} is a given threshold for dynamic point determination.

As a basic check for bad matches, any point that has a $b_i(n) \leqslant b_{bkgd}$ or $d_i(n) \geqslant d_{moving}$ is considered background or noise, respectively, and removed. The rest of the points are assumed to be accurately tracking the moving object or are classified as background using Eqs. (11) and (12).

The background and moving object thresholds are determined after testing the capabilities of the optical flow algorithm with the proposed method of optical flow background estimation (see Section 5).

The centroid of the moving points should roughly correlate with the object's center. This assumes a normal distribution of good features across the UAS. The mean of the current locations, $\mathbf{g}_i$, of the remaining points gives the centroid in pixels,

$$centroid = (x,y)_{pix} = \frac{1}{N}\sum_{n=1}^{N} g_i(n), \tag{13}$$

where N is the number of moving points. The mean of the current displacement vectors, $\mathbf{d}_i$, of the remaining points estimates the objects translation,

$$velocity = (x,y)_{vel} = \frac{1}{N}\sum_{n=1}^{N} d_i(n). \tag{14}$$

4.2. Kalman filter

The Kalman filter is used to predict the movement of the UAS. The Kalman filter model is given by

$$\begin{cases} \mathbf{X}_i = \mathbf{A}\,\mathbf{X}_{i-1} + \boldsymbol{\nu}_{i-1} \\ \mathbf{Z}_i = \mathbf{C}\,\mathbf{X}_i + \boldsymbol{\mu}_i \end{cases} \tag{15}$$

where $\mathbf{X}_i$ is the state vector, $\mathbf{Z}_i$ is the measurement vector, $\mathbf{A}$ is the state transition matrix, $\mathbf{C}$ is the measurement matrix, $\boldsymbol{\nu}_{i-1}$ is the state noise, and $\boldsymbol{\mu}_i$ represents the measurement noise. The state and measurement noise, $\boldsymbol{\nu}_{i-1}$ and $\boldsymbol{\mu}_i$, respectively, are assumed to be Gaussian random variables with zero mean. Their probability density functions are $N[0, \mathbf{Q}_{\mathbf{k_i}}]$ and $N[0, \mathbf{R}_{\mathbf{k_i}}]$, where the covariance matrix $\mathbf{Q}_{\mathbf{k_i}}$ and $\mathbf{R}_{\mathbf{k_i}}$ are referred to as the transition noise covariance matrix and measurement noise covariance matrix.

A constant acceleration system model is assumed with the model given by

$$\mathrm{X}_i = \mathrm{X}_{i-1} + \dot{\mathrm{X}}_{i-1}\Delta t + \frac{1}{2}\ddot{\mathrm{X}}_{i-1}\Delta t^2 \tag{16}$$

where Δt is discrete time and $\mathbf{X}$ represents two independent state vectors. Therefore, two separate Kalman filters are incorporated for each direction such that the state vectors look like $\mathbf{X}_{i,x} = [\mathrm{x}, \dot{\mathrm{x}}, \ddot{\mathrm{x}}]^\top$ and $\mathbf{X}_{i,y} = [\mathrm{y}, \dot{\mathrm{y}}, \ddot{\mathrm{y}}]^\top$. The resulting transition matrix is given in Eq. (17).

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

Both position and velocity are measured in a given image, I, so the measurement matrix is given in

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{18}$$

The noise covariance matrix, $\mathbf{Q}_{k_{i-1}}$, is the identity matrix multiplied by 1×10^{-4} and the measurement noise covariance, $\mathbf{R}_{\mathbf{k_i}}$, is the identity matrix multiplied by 1×10^{-3}.

4.3. Pan/tilt control

The goal of the pan/tilt controller is to track the object near the center of the image. If the object were not moving, then the pan/tilt unit would simply need to point at the previous location. Because the object is moving, the Kalman filter is used to predict the next location, so that the pan/tilt unit will point to the anticipated next location. Note that since the Kalman filter predicts the object location in pixels, it is independent of the pan/tilt controller, so long as the object remains in the image. Therefore, as long as the measurements, $(x,y)_{pix}$ and $(x,y)_{vel}$, are accurate, the system will be stable. The controller does control how abrupt the motion of the pan/tilt base is.

A deadzone is implemented based on an acceptable bound of the object in the image. If the predicted point is more than x_{dz} or y_{dz} away from the image center, then the pan/tilt is commanded to move the appropriate axis so that the predicted point will lie on the deadzone boundary for a scale factor of $k = 1$. The deadzone function is

$$dz(x, x_{dz}) = \begin{cases} k(x + x_{dz}) & x < -x_{dz} \\ 0 & |x| < x_{dz} \\ k(x - x_{dz}) & x \geqslant x_{dz} \end{cases} \tag{19}$$

for the x direction. The result of the deadzone calculation for each independent direction is then converted to pan/tilt units, added to the current position, and then sent to the pan/tilt controller:

$$\begin{aligned} x_{\text{pan direction}} &= dz(x, x_{dz})c_x + x_{\text{pan reference}} \\ y_{\text{tilt direction}} &= dz(y, y_{dz})c_y + y_{\text{tilt reference}} \end{aligned} \tag{20}$$

where c_x, c_y represent the conversion constants from pixels to pan/tilt units in the x and y directions, respectively.

4.4. Algorithm

A flowchart and steps for real-time tracking of a single, moving object are provided in Fig. 7 and Table 2, respectively. The system rate consists of the time required to capture an image, process the algorithm, and move the pan/tilt unit. Real-time tracking for this paper consists of image capture and algorithm processing speeds $\geqslant$30 fps and system rates $\geqslant$15 fps (i.e., image capture and algorithm processing speed $\leqslant 0.03\bar{3}$ ms and pan/tilt movement $\leqslant 0.03\bar{3}$ ms for a combined system rate $\leqslant 0.06\bar{6}$ ms).

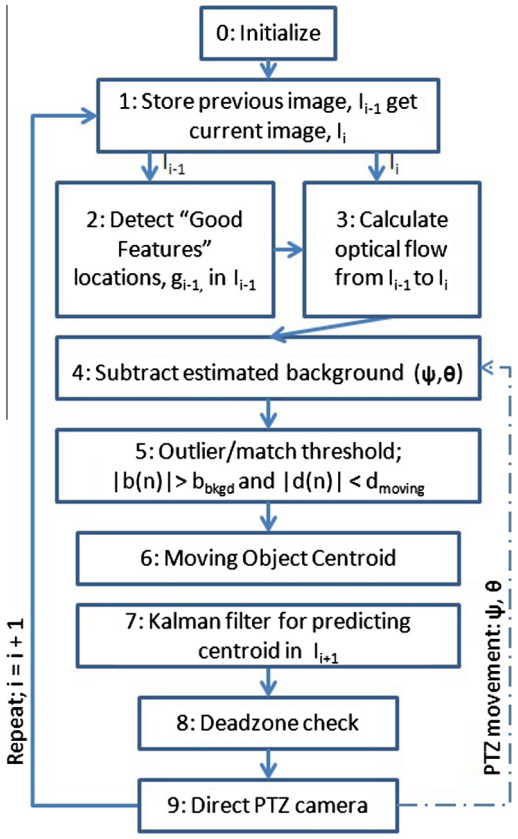

Fig. 7. Flowchart and steps for real-time tracking of a single, moving object.

5. Experiments and results

5.1. Point estimation performance

The experiments in this section use the closed-form solution of the *background motion*. Initial experiments contained a feature-filled background shown in Fig. 2.

It is assumed that the object is no closer than 1 m with a system rate of 15 fps. Using optical flow background estimation and the previous assumptions, Table 3 gives approximate pixel movements and object speed for a given pan or tilt angle, Θ, with focal length equal to 1076 pixels used to interpret scale of results.

An object moving 1.2 m/s at 1 m distance will travel 90 pixels at a system rate of 15 fps if the camera is not moving. As the camera tracks the object, the background points will move 90 pixels while the object will appear stationary. Because the threshold to identify erroneous matches is based on the midpoint of $\mathbf{g}_{i-1}$ and $\hat{\mathbf{g}}_i$, a minimum threshold would be 45 pixels and so this is used for d_{moving} (see Fig. 6).

Testing included the use of *only* the feature-filled background (see Fig. 2 for a sample of the background). Because all of the points are known to be static, any difference in optical flow points from *background motion* is an error in the method. Horizontal, vertical and diagonal camera movements at varying increments with one thousand features per frame were used. Table 4 shows the mean with standard deviation in parentheses for each test along with the number of frames used.

The large errors of some points unduly influenced the results of Table 4, so a classification system (see Section 4.1) was used to identify points that could be ruled out as incorrect matches.

Using the background estimation with the moving object threshold, Fig. 8 shows the percentage of features accounted for with respect to the estimated distance in pixels for a movement of 40 pixels.

Since larger errors are seen with diagonal movements, as shown in Fig. 8, a combination of ψ and θ movements is used as a means for understanding performance limitations of the system. Therefore, Fig. 9 shows the test scenario for diagonal movements with the number of features accounted for versus distance. Given $\psi \leqslant 1°$, $\theta \leqslant 1°$, 99% of the placed features are accounted for to within 4 pixels. Figs. 8 and 9 were used to come up with an appropriate background threshold, b_{bkgd}, to account for points based on selected movements. Matches further than

Table 2
Pan/tilt tracking with background subtraction.

Step	Procedure
0	Initialize parameters and capture first image: $(x,y)_{ref} = (0,0)$, $(x,y)_{rel} = (0,0)$, and image, I_0; $i = i + 1$
1	Store previous image, I_{i-1} and get current image I_i
2	Detect (x,y) coordinates of "good features", $\mathbf{g}_{i-1}$, in I_{i-1}
3	Calculate optical flow from I_{i-1} to I_i such that $\mathbf{g}_{i-1}$ gives $\mathbf{g}_i$
4	Convert relative pan/tilt units, $(x,y)_{rel}$, to ψ and θ and estimate background pixel movement, $\hat{\mathbf{g}}_i$, using $\mathbf{g}_{i-1}$ Subtract $\mathbf{g}_i$ from the midpoint of $\mathbf{g}_{i-1}$ and $\hat{\mathbf{g}}_i$ to give M displacements, $\mathbf{d}_i = \|\mathbf{g}_i - \frac{1}{2}(\mathbf{g}_{i-1} + \hat{\mathbf{g}}_i)\|$ Subtract $\hat{\mathbf{g}}_i$ from the optical flow movement to give N displacements, $\mathbf{b}_i = \|\mathbf{g}_i - \hat{\mathbf{g}}_i\|$
5	Points in $\mathbf{b}_i$: $\|b_i(n)\| > b_{bkgd}$ and $\mathbf{d}_i$: $\|d_i(n)\| < d_{moving}$ are considered moving $\forall n \in \{\mathbf{g}_i\}$
6	Determine the pixel centroid, $(x,y)_{pix}$ and magnitude, $m(x,y)$, of remaining n in $\mathbf{g}_i$
7	Convert $(x,y)_{pix}$ and $m(x,y)$ measurements to the pan/tilt coordinate system and use a Kalman filter for predicting $(x,y)_{rel}$ for I_{i+1}
8	Convert $(x,y)_{rel}$ to $(x,y)_{pix}$ and apply deadzone check functions $dz(x,x_{dz})$ and $dz(y,y_{dz})$ If $(x,y)_{pix}$ is within a rectangular deadzone, then $(x,y)_{ref}$ remains the same Otherwise, $(x,y)_{ref}$ is adjusted based upon the deadzone, the scale factor, k, and $(x,y)_{pix}$
9	Convert $(x,y)_{pix}$ to $(x,y)_{rel}$ Update the reference pan/tilt coordinates, $(x,y)_{ref} = (x,y)_{ref} + (x,y)_{rel}$, and direct the pan/tilt camera
10	Repeat starting at step 1

Table 3
Approximate pixel movement and speed for pan or tilt angle, Θ, with focal length = 1076 pixels.

Θ(°)	1.0	1.5	2.1	2.6	3.1	3.7	4.2	4.7
Pixel movement	20	30	40	50	60	70	80	90
Speed (m/s)	0.3	0.4	0.6	0.7	0.8	1.0	1.1	1.2

Note: Speed relates to an image plane 1 m from the camera with a system rate of 15 fps.

8 pixels away were very spread out, resulting in limited advantage of a b_{bkgd} larger than 8 pixels for the tested motions. Ideally, the smallest threshold should be used, since it creates a minimum speed that a moving object would be considered stationary. At 15 fps, 8 pixels would correspond to a speed of 0.1 m/s at 1 m.

Table 5 shows average algorithm processing times in milliseconds over 4000 iterations for varying number of features used and image resolution. A resolution of 656×524 with 250 features leaves 14.5 ms for camera and pan/tilt control for real-time tracking at 30 fps. The camera used for these experiments achieved an average time to capture and process an image of 3.4 ms for the requested image resolution of 656×524.

5.2. SUT tracking and timing

Experiments were conducted by swinging the SUT in the range of 30–40° from rest. This provides a maximum SUT velocity of 1.1–1.5 m/s. The complete system, consisting of image capture, algorithm, and pan/tilt movement; tracked the SUT at a system rate of 6–18 fps.

The SUT scenario shown in Fig. 10 shows the various workings of the algorithm for a moving SUT and static camera. Points that lie less than d_{min} are eliminated and shown as red, closed circles. In particular, it is seen that those points on the doorknob are correctly identified as background motion and are eliminated from the centroid calculation. The white, closed circle points show the matched optical flow points on the moving SUT for determining the pixel centroid and velocity vector.

The second scenario shows the algorithm at work for a moving SUT, moving camera (see Fig. 11). Again, those

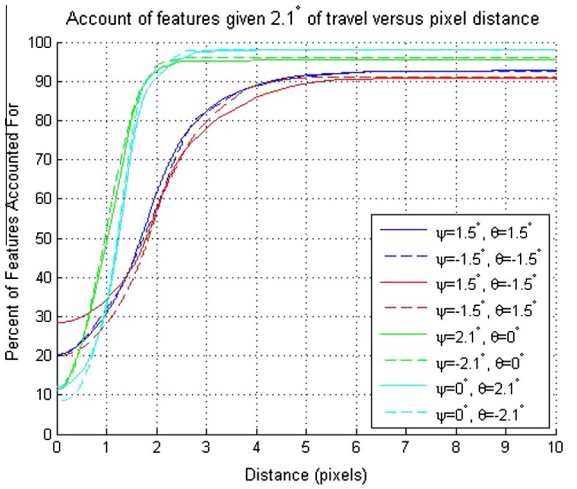

Fig. 8. Account of features (using background estimation with a moving object threshold) given equal degrees of travel versus pixel distance.

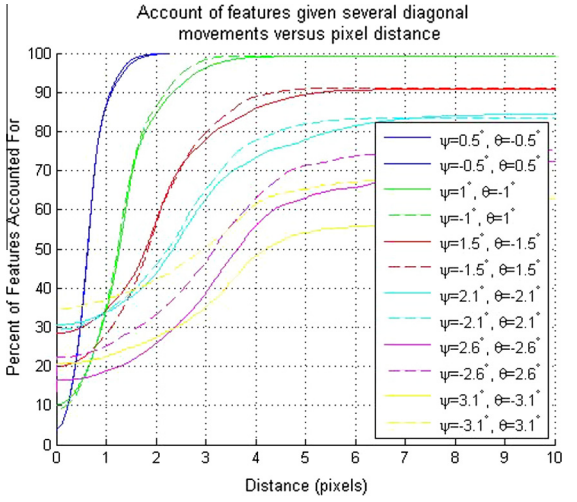

Fig. 9. Account of features (using background estimation with a moving object threshold) given a diagonal movement versus pixel distance. *Note:* a movement of $\psi = 3.1°$, $\theta = 3.1°$ is actually 4.4° of travel.

Table 4
Optical flow error for a stationary background with varying increments.

Θ (°)	Vertical $\psi = 0$, $\theta = \Theta$ x, y pixels (x, y pixels)	Horizontal $\psi = \Theta$, $\theta = 0$ x, y pixels (x, y pixels)	Diagonal $\psi = \Theta$, $\theta = \Theta$ x, y pixels (x, y pixels)	Diagonal $\psi = \Theta$, $\theta = -\Theta$ x, y pixels (x, y pixels)	Frames (#)
−3.1	−3.1, 17.8 (37.0, 39.2)	−47.1, 6.3 (51.6, 36.8)	−49.8, 28.6 (68.9, 55.7)	−36.9, −25.6 (53.4, 51.8)	16
−2.6	−1.3, 9.5 (23.9, 30.1)	−33.8, 7.0 (43.3, 34.2)	−35.8, 23.7 (60.1, 49.8)	−26.4, −17.0 (48.1, 43.0)	19
−2.1	−0.8, 3.5 (20.2, 19.1)	−10.5, 1.7 (33.4, 22.7)	−20.8, 13.6 (45.7, 40.9)	−18.1, −11.3 (38.6, 34.9)	24
−1.5	−0.4, 0.8 (11.1, 15.0)	−2.3, −0.3 (21.5, 12.3)	−5.1, 2.8 (25.0, 22.3)	−6.7, −4.8 (24.3, 23.3)	33
−1.0	−0.7, −0.7 (13.1, 2.2)	0.0, −0.4 (12.0, 8.6)	−0.6, −0.7 (13.0, 8.1)	−0.8, −0.1 (13.8, 12.5)	49
−0.5	−0.1, −0.4 (3.9, 1.2)	0.2, −0.1 (5.5, 4.1)	−0.1, −0.3 (3.9, 1.9)	−0.1, 0.2 (5.9, 5.5)	99
0.5	0.5, −0.2 (6.3, 6.4)	−0.3, 0.0 (0.6, 1.7)	−0.1, 0.1 (6.5, 6.6)	0.0, −0.4 (2.7, 1.7)	99
1.0	−0.9, 0.0 (15.4, 12.3)	−0.9, −0.5 (9.8, 9.7)	−0.3, −0.3 (12.5, 13.7)	0.0, −0.9 (10.8, 8.5)	49
1.5	−0.7, −0.8 (14.7, 14.6)	2.2, −0.7 (16.5, 14.2)	6.9, −5.2 (25.0, 23.6)	7.2, 3.9 (25.4, 26.0)	33
2.1	−1.6, −3.7 (22.2, 23.3)	7.4, −0.7 (21.7, 18.4)	15.6, −16.0 (39.6, 40.2)	15.3, 9.5 (30.7, 31.7)	24
2.6	−1.7, −11.2 (23.3, 34.5)	32.5, 2.8 (35.4, 33.3)	31.2, −25.0 (49.9, 48.6)	23.2, 16.2 (36.9, 37.8)	19
3.1	−2.6, −22.2 (33.4, 44.7)	42.8, 2.6 (38.5, 36.2)	41.5, −31.3 (56.2, 53.9)	34.5, 23.2 (50.5, 47.8)	16

Note: x, y pixels (x, y pixels) are the global mean and standard deviations, respectively.

Table 5
Average algorithm time (in milliseconds) for varying image resolutions versus number of features used.

	Image resolution (width × height)			
# Of features	328 × 262	656 × 524	984 × 786	1312 × 1048
250	14.0	18.8	21.5	30.6
500	19.6	24.1	27.3	38.0
750	22.1	30.9	33.3	45.8
1000	21.8	36.4	39.4	52.0

points near the doorknob are eliminated as they are identified as stationary, global points. Despite movement of approximately 49 pixels, the red points near the doorknob in Fig. 11 are still eliminated *background motion*.

Fig. 12 provides a sequence of frames, approximately 0.24 seconds apart for one period, showing the tracking of the moving (i.e., swinging) SUT with a moving camera. Note that both upper and lower, left images show eliminated points in red due to the lack of motion of the SUT during transition.

Timing of the processes involved were determined using the CPU clock. An average of 999 iterations for image capture, algorithm, and pan/tilt movement is shown in Table 6.

The results show that a system rate of 6–18 fps per camera may be realized for the entire process. The slowest part of the process consisted of directing the pan/tilt unit at a minimum of 43 ms and a maximum of 159 ms for the test given. Larger movements caused larger delays in pan/tilt movement in order for the unit to complete the directed motion. The algorithm operated at a minimum of 10 ms and a maximum of 17 ms, and the camera operated at a minimum of 3.2 ms and a maximum of 3.4 ms for a 656 × 524 image. Faster processing of 21 fps is possible by not waiting for the pan/tilt unit to complete directed tasks, but the error would not be compensated for. This would require tuning based on the lag time for a given pan/tilt unit.

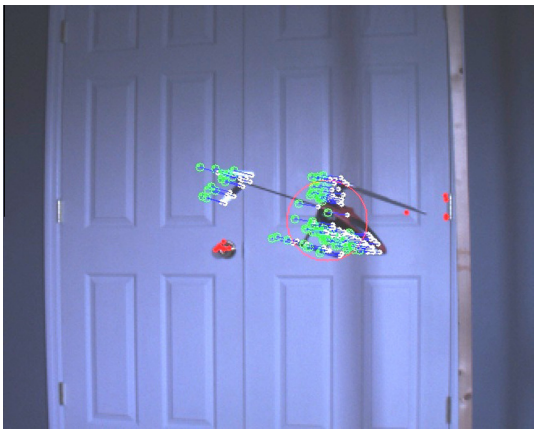

Fig. 10. A moving SUT captured by a static camera with previous points shown as green rectangles, $\mathbf{g}_{i-1}$, current optical flow points as white, closed circles, $\mathbf{g}_i$, estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

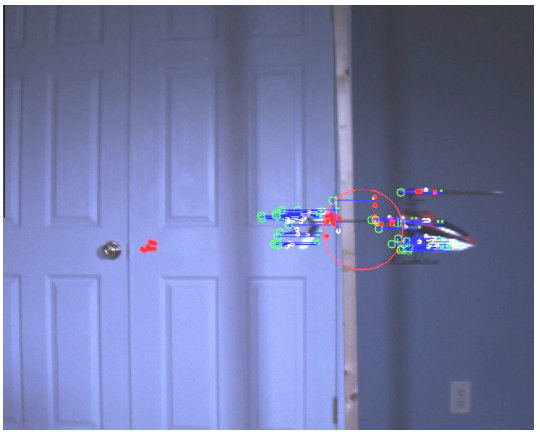

Fig. 11. A moving SUT captured by a moving camera with previous points shown as green rectangles, $\mathbf{g}_{i-1}$, current optical flow points as white, closed circles, $\mathbf{g}_i$, estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3. SUT comparison testing

The proposed algorithm requires no training or matching of features to templates. It is believed that the technique used in this paper captures faster, single moving object's better than existing general purpose methods. Training for specialized tracking is required in [13,18,27]. A color histogram representation is required in [26,28,29] and often fails with large shifts in motion or objects of similar colors. Specialized tracking of skin color for face detection is used in [23]. It is sought here to provide further comparisons in order to give a general idea of realizable performance characteristics by other algorithms with generic assumptions.

Comparisons were made using a set of 999 images captured using the optical flow background estimation. The timing between images corresponds with Table 6 and accuracy is determined through a combination of theoretical pendulum placement and user verification.

The comparison algorithms chosen were accessible via the OpenCV library [31]. The chosen comparison algorithms are optical flow, Color-Adaptive-Meanshift (CAMShift) and GPU SURF with Fast Library of Approximate Nearest Neighbors (FLANN) matching. Since these algorithms vary in user requirements, tuning and utility, it is sought here to provide the reader a general idea of performance characteristics and comparability of the techniques developed in this paper and not, necessarily, optimized techniques.

Merely using optical flow requires the user to specify how many points to track and where to start. Due to large shifts in the background, optical flow points are quickly lost and re-selection is required to continue tracking. Therefore, it was sought to correct these large displacements due to camera movement using phase correlation

Fig. 12. A sequence of frames, approximately 0.24 s apart for one period, of a swinging SUT with the center of the image identified by a white diamond, previous points shown as green rectangles, $\mathbf{g}_{i-1}$, current optical flow points as white, closed circles, $\mathbf{g}_i$, estimated points as green, open circles, $\hat{\mathbf{g}}_i$, and removed outliers shown as red, closed circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Timing of image capture, algorithm and pan/tilt movement over 999 iterations.

Procedure	Average (ms)	Minimum (ms)	Maximum (ms)
Image capture	3.4	3.2	3.4
Algorithm	12.7	10	17
Pan/Tilt movement	65.0	43	159
Total	81.0	56.2	179.4

(e.g., phase correlation was used in [14] to provide a rough estimate of displacement). However, the lack of background features and the size of the moving object in the given test was detrimental in maintaining optical flow points. Therefore, phase correlation was not used in the experiments. Both the moving object size and background features are critical in using phase correlation.

The CAMShift algorithm generally tracked well at slower speeds where the object was within the range of a user-specified kernel. Tracking ceased where the local maximum was found to be a background point that eventually left the screen upon pan/tilt movement. Using a background with the same color as the moving object of interest would most likely degrade the tracking of the object and would require user re-selection of the area of interest (see Fig. 13).

The GPU SURF feature detection was used due to it's speed along with matching using the Fast Library for Approximate Nearest Neighbors (FLANN). System timing, using solely the GPU SURF detection algorithm without matching, provided an average processing speed of around 45 ms. Note that this time is already greater than that achieved in [13] using OpenCL for the entire process. Therefore, in fairness, the process could be optimized to possibly perform better tracking and timing than given here. Training images consisted of twenty images with varying camera parameters of the helicopter on a white background as shown in Fig. 14 (right). Matches were rank ordered based upon a minimum distance associated with a match. Matches within 20% of the best match were used to determine the moving object's centroid. From preliminary testing, the threshold of 20% was used subjectively to eliminate outliers based upon examination of match distances for the given test. The closest match was not always

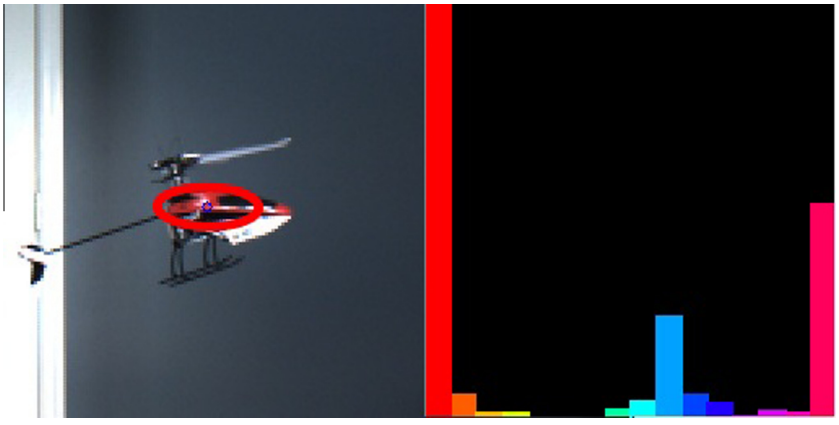

Fig. 13. For comparison purposes, the CAMShift algorithm was used to determine the object centroid. The color histogram (right) is made up of the components within the ellipse of an image in the test set (left). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

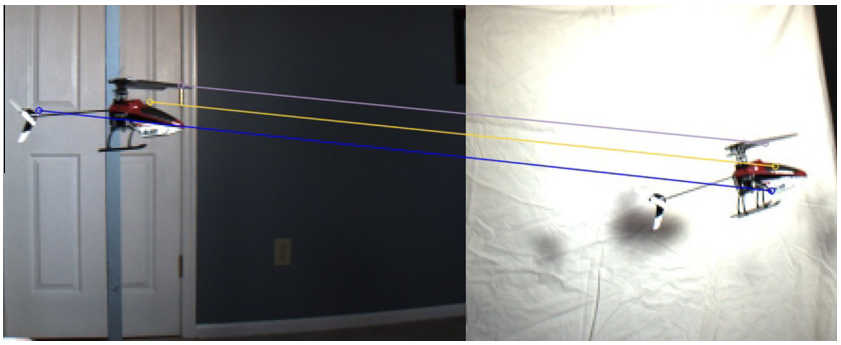

Fig. 14. For comparison purposes, good matches were used to provide the moving object centroid using a point average. The training image (right) shows feature locations that were matched to an image in the test set (left).

Table 7
General performance characteristics of object detection/tracking systems for a set of 999 images at 656 × 524 resolution.

Algorithm	Training/Selection	Timing/(System rate)	Pixel error/(# re-selects)
GPU Optical Flow Background Estimation	–/–	12.7 ms/(12 fps)	36 ± 34/–
Optical flow	–/Selection	22.4 ms/(11 fps)	60 ± 65/(46)
CAMShift	–/Selection	22.5 ms/(11 fps)	43 ± 95/(23)
GPU SURF	Training/–	56.4 ms/(8 fps)	56 ± 51/–

Note: System timing includes the average times for image capture and pan/tilt operation.

found to be on the moving object and therefore false matches were often identified.

An advantage of the technique described in this paper is that it tracks a single, moving object despite changing features and background while others require similarity in features between frames. For example, the proposed algorithm would be able to track a UAS from any orientation regardless of features whereas feature-based trackers require training or minimal transient feature characteristics.

Table 7 shows some performance characteristics of the technique developed in this paper (GPU optical flow background estimation) along with the selected algorithms. Again, these results provide a general idea of performance characteristics related to those of the comparisons described in Section 1.2.

6. Conclusions

A real-time tracking algorithm capable of 30 fps using GPU-based algorithms for a pan/tilt camera was presented. The theory and approximations were provided to estimate *background motion* for optical flow background subtraction. The experiments with the static background showed minute benefit in classifying points further than 8 pixels from the predicted location as background. The larger this threshold, the larger the lower limit on the speed of the moving object. Actual tracking experiments showed that greater angles, $\geqslant 4.4^{\circ}$, could be achieved by tracking a helicopter at speeds of $\geqslant 1.2$ m/s with the system operating between 6 and 18 fps.

An algorithm was developed consisting of point classification, Kalman filtering and pan/tilt control where point classification consisted of whether points were background, motion or noise. The magnitude of the camera movement was varied to show accuracy of the estimation of the background motion by comparing it with the measured optical flow. Results show algorithm processing times for various image resolutions and number of features which shows how performance can scale to achieve desired frame rates. In addition, the results show that the technique developed in this paper requires no training, initialization/operator selection, or pre-defined capture area (i.e., alert zone). The novel combination captures faster, single moving object's better than existing general purpose methods.

Acknowledgment

We would like to thank the Air Force Research Laboratory and the Air Force Office of Scientific Research for funding support.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.measurement.2013.10.025.

References

[1] USAF Unmanned Aircraft Systems Flight Plan 2009-2047, Tech. Rep., USAF, 2009.
[2] D.B. Doman, M.W. Oppenheimer, D.O. Sigthorsson, Dynamics and control of a biomimetic vehicle using biased wingbeat forcing functions: Part II – controller, in: 48th AIAA Aerospace Sciences Meeting, Orlando, FL, 2010, http://dx.doi.org/10.2514/1.49735.
[3] M.L. Anderson, Design and Control of Flapping Wing Micro Air Vehicles, Ph.D. Thesis, Air Force Institute of Technology, 2011, <http://www.dtic.mil/docs/citations/ADA549053>.

[4] A. Jennings, J. Black, Texture-based photogrammetry accuracy on curved surfaces, AIAA J. 50 (2012) 1060–1071, http://dx.doi.org/10.2514/1.J050956.
[5] R. Sahai, K. Galloway, R. Wood, Elastic element integration for improved flapping-wing micro air vehicle performance, IEEE Trans. Robot. 29 (1) (2013) 32–41, http://dx.doi.org/10.1109/TRO.2012.2218936.
[6] J.-S. Lee, J.-H. Han, Experimental study on the flight dynamics of a bioinspired ornithopter: free flight testing and wind tunnel testing, Smart Mater. Struct. 21 (9) (2012) 094023, http://dx.doi.org/10.1088/0964-1726/21/9/094023.
[7] J. Ratti, G. Vachtsevanos, A biologically-inspired micro aerial vehicle, J. Intell. Robot. Syst. 60 (1) (2013) 153–178, http://dx.doi.org/10.1007/s10846-010-9415-x.
[8] D. Magree, A Photogrammetry-Based Hybrid System for Dynamic Tracking and Measurement, Master's Thesis, Air Force Institute of Technology, 2010, <http://www.dtic.mil/docs/citations/ADA523522>.
[9] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (4) (2006) 1–45, http://dx.doi.org/10.1145/1177352.1177355.
[10] M. Smith, A. Boxerbaum, G. Peterson, R. Quinn, Electronic image stabilization using optical flow with inertial fusion, in: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 2010, pp. 1146–1153, http://dx.doi.org/10.1109/IROS.2010.5651113.
[11] A. Bevilacqua, P. Azzari, High-quality real time motion detection using PTZ cameras, in: IEEE Intl. Conference on Video and Signal Based Surveillance AVSS '06, 2006, pp. 23, http://dx.doi.org/10.1109/AVSS.2006.57.
[12] X. Clady, F. Collange, F. Jurie, P. Martinet, Tracking with a pan-tilt-zoom camera for an ACC system, in: 12th Scandinavian Conference on Image Analysis (SCIA '01), Bergen, Norway, 2001, pp. 561–566.
[13] J. Cao, X. Xie, J. Liang, D. Li, GPU accelerated target tracking method, in: D. Jin, S. Lin (Eds.), Advances in Multimedia, Software Engineering and Computing, Advances in Intelligent and Soft Computing, vol. 1, 128, Springer, Berlin, Heidelberg, 2012, pp. 251–257.
[14] P. Azzari, L. Di Stefano, A. Bevilacqua, An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera, in: IEEE Conference on Advanced Video and Signal Based Surveillance, 2005, AVSS, 2005, pp. 511–516, http://dx.doi.org/10.1109/AVSS.2005.1577321.
[15] S. Kang, J.-K. Paik, A. Koschan, B.R. Abidi, M.A. Abidi, Real-time video tracking using PTZ cameras, in: K.W. Tobin Jr., F. Meriaudeau (Eds.), Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 5132, 2003, pp. 103–111, http://dx.doi.org/10.1117/12.514945.
[16] K. Kanatani, Camera rotation invariance of image characteristics, in: Computer Vision, Graphics, and Image Processing, vol. 39, 1987, pp. 328–354, http://dx.doi.org/10.1016/S0734-189X(87)80185-8.
[17] D. Murray, A. Basu, Motion tracking with an active camera, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, 1994, pp. 449–459, http://dx.doi.org/10.1109/34.291452.
[18] X. Clady, F. Collange, F. Jurie, P. Martinet, Object tracking with a pan-tilt-zoom camera: application to car driving assistance, in: Proceedings 2001 ICRA. IEEE Intl. Conference on Robotics and Automation, vol. 2, 2001, pp. 1653–1658, http://dx.doi.org/10.1109/ROBOT.2001.932848.
[19] Y. Ma, S. Soatto, J. Kosecka, S.S. Sastry, An Invitation to 3D Vision, Springer, 2003.
[20] I. Grinias, G. Tziritas, Robust pan, tilt and zoom estimation, in: Proc. 14th Int. Conf. Digital Signal Processing DSP 2002, vol. 2, 2002, pp. 679–682, http://dx.doi.org/10.1109/ICDSP.2002.1028182.
[21] B. Shirmohammadi, C.J. Taylor, Distributed target tracking using self localizing smart camera networks, in: ACM Intl. Conf. on Distributed Smart Cameras, 2010, pp. 16–24, http://dx.doi.org/10.1145/1865987.1865991.
[22] P. Guha, D. Palai, D. Goswami, A. Mukerjee, Dynatracker: Target tracking in active video surveillance systems, in: Proceedings., 12th Intl. Conference on Advanced Robotics, 2005, ICAR '05, 2005, pp. 621–627, http://dx.doi.org/10.1109/ICAR.2005.1507473.
[23] M. Al Haj, A. Bagdanov, J. Gonza'lez, F. Roca, Reactive object tracking with a single PTZ camera, in: 20th Intl. Conference on Pattern Recognition (ICPR), 2010, pp. 1690–1693, http://dx.doi.org/10.1109/ICPR.2010.418.
[24] C.-S. Yang, R.-H. Chen, C.-Y. Lee, S.-J. Lin, PTZ camera based position tracking in IP-surveillance system, in: 3rd Intl. Conference on Sensing Technology, ICST, 2008, pp. 142–146, http://dx.doi.org/10.1109/ICSENST.2008.4757089.
[25] T. Hoedl, D. Brandt, U. Soergel, M. Wiggenhagen, Real-time orientation of a PTZ-camera based on pedestrian detection in video data of wide and complex scenes, XXI ISPRS Congress 37 (3b) (2008) 663–668, <http://www.isprs.org/proceedings/XXXVII/congress/3b_pdf/32.pdf>.
[26] P. Kumar, A. Dick, T.S. Sheng, Real time target tracking with pan tilt zoom camera, in: Proc. DICTA '09. Digital Image Computing: Techniques and Applications, 2009, pp. 492–497, http://dx.doi.org/10.1109/DICTA.2009.84.
[27] M.A. Akhloufi, Pan and Tilt Real-Time Target Tracking, SPIE, vol. 7668, 2010, pp. 76680K-1–10, http://dx.doi.org/10.1117/12.849766.
[28] H. Zhang, J. Hong, W. Lin, L. Li, Design of tracking system based on mean-shift and Kalman filter, in: MIPPR 2009: Automatic Target Recognition and Image Analysis, 2009, http://dx.doi.org/10.1117/12.832551.
[29] Y. Li, Q. Xie, G. Yu, Real-time tracking system combining mixture of gaussian model and CAMshift algorithm, in: Intl. Conference on Image Processing and Pattern Recognition in Industrial Engineering, 2010, http://dx.doi.org/10.1117/12.867102.
[30] Z. Qigui, L. Bo, Search on automatic target tracking based on PTZ system, in: 2011 Intl. Conference on Image Analysis and Signal Processing (IASP), 2011, pp. 192–195, http://dx.doi.org/10.1109/IASP.2011.6109027.
[31] G. Bradski, The OpenCV Library, 2000, <http://opencv.org>.