Low-Complexity Global Motion Estimation for Aerial Vehicles

Nirmala Ramakrishnan, Alok Prakash, Thambipillai Srikanthan
Nanyang Technological University, Singapore
rama0056@e.ntu.edu.sg (alok, astsrikan)@ntu.edu.sg

Abstract

Global motion estimation (GME) algorithms are typically employed on aerial videos captured by on-board UAV cameras, to compensate for the artificial motion induced in these video frames due to camera motion. However, existing methods for GME have high computational complexity and are therefore not suitable for on-board processing in UAVs with limited computing capabilities. In this paper, we propose a novel low complexity technique for GME that exploits the characteristics of aerial videos to only employ the minimum, yet, well-distributed features based on the scene complexity. Experiments performed on a mobile SoC platform, similar to the ones used in UAVs, confirm that the proposed technique achieves a speedup in execution time of over 40% without compromising the accuracy of the GME step when compared to a conventional method.

1. Introduction

Unmanned aerial vehicles (UAVs) are increasingly being deployed for wide area monitoring and surveillance to collect high resolution video imagery of the area that could otherwise be remote and inaccessible [25]. These battery-powered UAVs operate on very stringent power and weight budgets that determine not only their flight duration but also the on-board processing and communication capabilities. The limited on-board processing capability and the need for achieving real-time surveillance, necessitates that the video data be expeditiously relayed to a ground station to extract useful information, such as the presence of moving objects, since they represent *interesting* events in surveillance [1]. However, the limited communication bandwidth itself may not allow high resolution videos to be transmitted. Therefore, low-complexity video processing algorithms, which can run effectively on the limited on-board processing systems, are essential to extract relevant information on-board and communicate only this information, in real-time [12].

Several computer vision techniques for motion detection have been proposed in the existing literature that separate moving objects from the scene [24]. This has also enabled efficient compression techniques such as region-of-interest coding of aerial videos, by compressing the moving objects separately from the background scene [16]. However, these algorithms are highly complex in terms of computations, especially due to the complexity in the global motion estimation (GME) step, which is one of the critical and time consuming steps in these algorithms [16]. GME is typically the first step in processing aerial videos captured from moving cameras, since it removes the artificial motion induced in the video frames, due to camera motion.

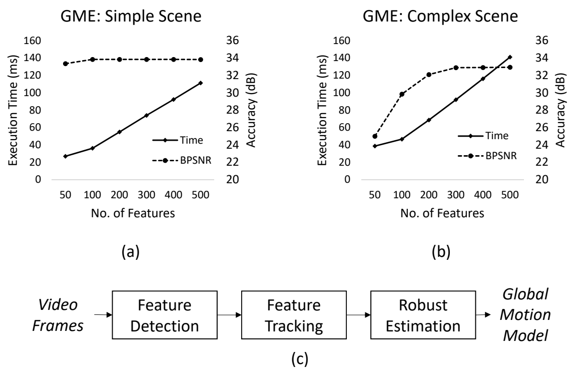


Figure 1. Feature-based GME: Execution time and accuracy for (a) Simple scene condition (b) Complex scene condition, and (c) Overall pipeline

Figures 1 (a) and 1 (b) show the execution time and accuracy results of the GME step for two scenarios. Figure 1 (a) shows the results when GME is applied on a video with few moving objects and simple translation motion, whereas Fig. 1 (b) shows the results for a video with large number of moving objects and a global rotation. As shown in Fig. 1 (c), the GME step operates on features obtained from the current frame, using interest point detectors such as Shi-Tomasi [22]. It can be observed from Fig. 1 (a) and 1 (b) that while the execution time increases significantly with increasing number of features in both scenarios, accuracy only improves in Fig. 1 (b), for the scene with large number of moving objects. Therefore, in order to keep the execution time (and hence the complexity) of the GME step low, it is prudent to use less number of features for simpler scene conditions (e.g. smaller number of moving objects) and intelligently ramp up the number of features for complex scenes, e.g. with large number of moving objects.

In this paper, we exploit this behavior to propose *sparse-GME*, a low-complexity GME method that employs only essential number of features depending on scene conditions.

The main contributions of this work are:

- A simple online evaluation strategy to detect successful completion of GME.
- A repopulation strategy to intelligently ramp up the number of features, in case GME fails with lower number of features.
- In addition to evaluating the proposed methods on a PC, as done in most of the existing work, we have also evaluated and proved the effectiveness of the proposed techniques on an ARM processor-based mobile embedded development platform, Odroid XU4 [3], to closely replicate [2][21] the on-board processing systems in UAVs.

2. Related Work

Feature-based GME methods, as shown in Fig. 1 (c), are preferred for their lower complexity. In this method, features are first detected in the current frame using interest point detectors such as Shi-Tomasi [22]. Next, their corresponding location in the previous frame is determined using a feature tracker such as the Lucas-Kanade (KLT) algorithm [14]. Finally, these feature correspondences are robustly fit to a global motion model using a robust estimator such as RanSaC [9].

In literature, low-complexity methods have been proposed for each of the stages in the GME pipeline. In feature detection, the cost of windowing operations is reduced in [15] and pruning methods have been used [19]. Lower-precision computations have been employed in [17] for low-complexity KLT. The complexity of RanSaC has been addressed through efforts for faster convergence: with early rejection of bad hypotheses [6], ordering feature correspondence based on their quality [7] and simultaneous ranking of multiple hypotheses [18]. However, all these methods still assume that a large number of feature correspondences is available, and therefore incur the associated computation overhead of tracking for a large feature set. Therefore, the proposed GME technique only employs a minimal number of well-distributed features for most of the time and intelligently ramps up the number of features if and when the scene becomes complex.

In [10], a light-weight feature matcher has been proposed to remove the camera motions in HDR photography. The authors effectively exploit the minimal translation between the successive images taken to generate an HDR image to

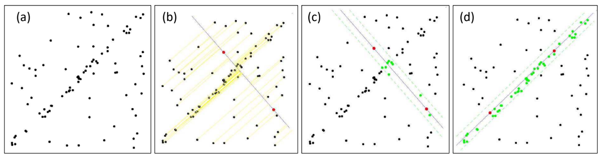


Figure 2. Illustration of Random Sample Consensus (RanSaC) algorithm [4] (a) Data points with noise (b) Compute error for a hypothesis set (c) Determine the corresponding consensus set (d) Find and report the largest consensus set after several iterations

lower the complexity of their feature-based image registration pipeline. However, in the context of aerial video surveillance, a much wider range of displacement is encountered due to varying vehicle speed and motion. Hence, a more robust feature tracker such as KLT [14] is essential, which while being more complex than the one used in [10], can effectively deal with wider translations to generate accurate feature correspondences. Hence, in this paper the proposed method employs a KLT based tracker on an intelligently selected set of features to ensure high tracking accuracy while still reducing the overall complexity.

3. Background

GME computes the parameters of a global (camera) motion model that defines the camera motion between successive frames I and J. For aerial videos, a homography model H is widely used [24] that associates every pixel at (x, y) in I to a new location (u, v) in J as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (1)$$

For each feature $X^I = (x, y)$ in frame I, its corresponding location in frame J, given by $X^J = (x', y')$ is determined by the feature tracking step. A set of N feature correspondences are used to solve for the parameters of H in a least-squared manner where the error E, between the measured correspondence (x'_i, y'_i) and the estimated correspondence (u_i, v_i), is minimized:

$$E = \sum_{i}^{N} (x'_i - u_i)^2 + (y'_i - v_i)^2 \qquad (2)$$

Feature correspondences can be noisy due to errors in feature localization and tracking. This is dealt with a large number (N) of feature correspondences that leads to higher accuracy in the estimation, in the presence of noise.

In addition, feature correspondences may belong to moving objects, and therefore need to be excluded as *outliers* from the estimation. Random Sample Consensus (RanSaC) [9] is a classic robust estimation algorithm that can deal with a large number of *outliers*. As shown in Fig. 2, it selects a random sample set and estimates the model (in this case a straight line) for this sample. It then computes the error, also known as reprojection error, of all data points to this model. By applying a threshold on the reprojection error as in Fig. 2 (b)(c), it generates a consensus set containing *inliers*. This process is used iteratively, until the probability of finding a consensus set larger than what has already been found is very small. The largest consensus set is reported as the *inliers*, as shown in Fig. 2 (d). With a large number of feature correspondences, it is expected that there are sufficient feature correspondences adhering to the global motion (in other words, belonging to the background) and they become *inliers* during RanSaC, thereby making all features that belong to the foreground or are too noisy, as *outliers*.

Aerial surveillance videos, captured by on-board cameras on UAVs, have the following characteristics that are of interest during GME:

1. *Small foreground*: The background occupies the majority of the scene and moving objects are sparse and present only occasionally in surveillance videos [13]. This implies that the outlier ratio can be expected to be fairly small for most of the video frames in the aerial video.

2. *Predominantly simple camera motion*: The camera motion experienced by successive frames in aerial surveillance videos is predominantly translational. Large rotations are only experienced when the vehicle performs bank turns [25]. This implies that the feature tracker can be setup for a smooth translation motion, and be expected to provide highly accurate feature correspondences for most of the video sequence. This, in turn, implies that the noise in the data used for GME can be expected to be low for most of the video frames in an aerial video.

Therefore, when the outlier ratio as well as the overall noise in the data is expected to be small, employing a large number of features, leads to wasted computations for the GME. By a careful selection of features, even a sparse set can achieve high quality GME for most of the video frames. It is proposed that the number of features should be adaptive to the scene content and camera motion, such that when scene conditions are conducive, the GME can be performed with sparse features and only when needed, higher density of features is employed.

4. GME with Sparse Features

The objective of the proposed sparse-GME method is to adapt to the complexity of the scene, in terms of the number of features used for GME, such that minimal number of features are employed for GME. The key idea is to begin with a careful selection of a sparse set of features, that works well

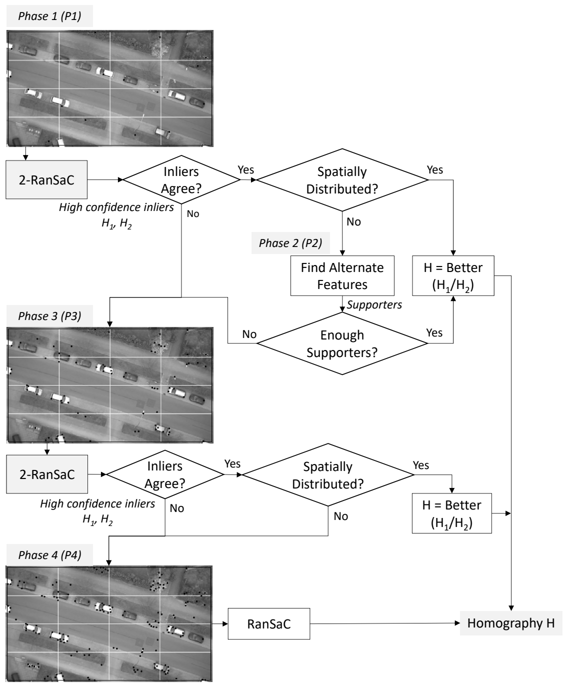


Figure 3. Flow of proposed *sparse-GME* method

for simple scene conditions, i.e. small number of moving objects (small outlier ratio) and translational camera motion (small noise in feature correspondences). GME is predominantly performed with this sparse set and only if it fails, then we intelligently ramp up the features to improve its accuracy.

Spatial distribution of the detected features is a critical criterion for the usefulness of feature detectors [8, 5]. Hence, we use this criterion to select sparse features, as it reduces the redundancy in representing local motion when camera motion is simple, and also ensures that not all the features fall on a few moving objects. Therefore, we propose to select a sparse feature set by choosing features that are well spread out in the frame. As shown in Fig. 3, the frame is processed in a block-based manner, extracting the best feature in each block of the grid and using this feature correspondence to represent the local motion in that block.

An *evaluation* method is proposed to determine if the GME can be successfully completed with the sparse feature set. To cater for situations where the GME cannot be completed successfully with the sparse feature set, we also proposed a *repopulation* strategy that systematically injects additional features to improve the accuracy of the GME. This leads to an iterative GME process as shown in Fig. 3. These steps are described in detail in following subsections.

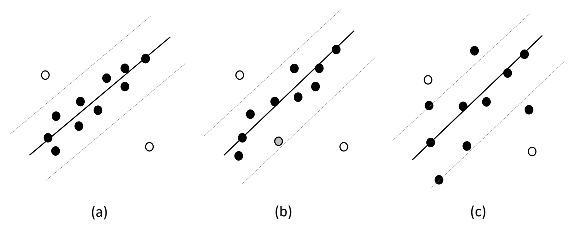


Figure 4. Illustration of RanSaC outcomes with sparse features: (a) Successful estimation with clear classification of inliers and outliers (b) Failed estimation with a single feature correspondence (grey) skewing the estimation (c) Failed estimation with very poor agreement between the inliers

4.1. Online Evaluation of GME

We propose a simple low-complexity strategy to evaluate the GME, which can be used to determine whether another iteration with additional features is needed to successfully complete the GME.

Figure 4 shows the possible outcomes of GME with sparse features by illustrating the feature correspondences in the model parameter space (as was described in Fig. 2). Figure 4 (a) is an example of a successful estimation: the inlier feature correspondences (black circles) are close to the fitted model (bold line), and the outliers (empty circles) are distinctly far from the inliers set. In this case, the RanSaC algorithm is able to correctly classify the inliers and outliers, and the model fitted by the inliers represents global motion.

Failure in estimation, i.e. when the estimated model does not represent the actual global motion experienced, is due to the following reasons:

1. Inaccurate correspondences or foreground features participate and *skew* the estimation. This means that the inlier/outlier classification itself is faulty. Two possible scenarios arise in this case:
    (a) As shown in Fig. 4 (b), majority of the correspondences are useful, however a very small number of inaccurate correspondences or features on moving objects (represented by grey circles) are responsible for erroneous estimation. Removing or replacing these feature correspondences could still lead to correct estimation.
    (b) As shown in Fig. 4 (c), majority of the correspondences are not agreeing well to the model. Therefore a higher density of features is needed so that the camera motion can be captured with sufficient number of features representing the global motion.
2. Inlier/outlier classification is correct, however the inliers form a *degenerate* set due to insufficient coverage of the frame.

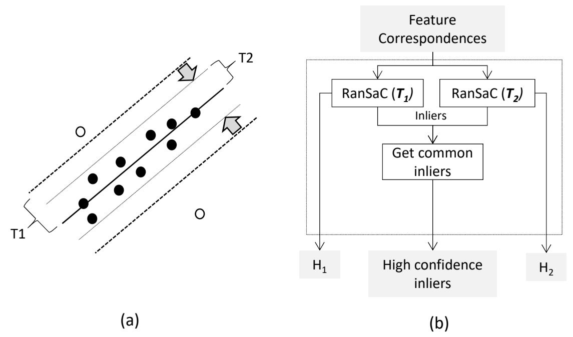


Figure 5. Proposed *2-RanSaC* strategy to select *high-confidence inliers*: Apply RanSaC with two reprojection thresholds, T_1 and T_2, with $T_2 < T_1$

The aim of the evaluation strategy is to distinguish between these outcomes, so that additional features are employed for GME, only when the current estimation has failed. The proposed evaluation strategy involves the identification of features that agree with the estimated model well, referred to as *high-confidence inliers*. We then apply two criteria: (1) Inlier agreement (2) Spatial distribution constraint, on these features to detect GME success/failure.

4.1.1 High-Confidence Inliers

In order to evaluate the accuracy of the estimation, the first step is to determine if the set of feature correspondences tightly agree with the estimated model. In RanSaC, the reprojection threshold, used to separate the inliers from the outliers, determines how tightly the inlier set agrees with the estimated model. We propose to invoke RanSaC with two reprojection thresholds, T_1 and T_2 such that $T_2 < T_1$, called *2-RanSaC* as shown in Fig. 5. If the inlier/outlier classification remains the same for both T_1 and T_2, then the inliers can be relied upon for further evaluation. The correspondences that are inliers for both the estimations are considered as *high-confidence inliers*. The correspondences that change their inlier/outlier status when the threshold is tightened are considered *shaky* correspondences.

4.1.2 Inlier Agreement

Next, we ensure that the inliers strongly agree to the model. The location error, LE of the inliers is computed and checked if it is below a threshold as follows:

$$LE = |H_{est}(x_i, y_i) - tracker(x_i, y_i)| \tag{3}$$
$$LE < T_{LE} \tag{4}$$

where for feature at (x_i, y_i), $H_{est}(x_i, y_i)$ represents the GM estimated location by applying the homography H_{est}

and $tracker(x_i, y_i)$ represents the tracked location. All the *high-confidence inliers* need to have a low location error. Successful estimation as in Fig. 4 (a) should first meet the above inlier agreement criterion.

4.1.3 Spatial Distribution Constraint

An additional check for the spatial distribution of the *high-confidence inliers* is proposed to rule out the possibility of a *degenerate* inlier set - i.e. the scenario when inliers agree well to the estimated model, but do not represent global motion due to lack of coverage of the frame. Although the frame coverage is ensured during feature detection, features may be lost during tracking or as outliers during estimation, resulting in poor coverage of inliers after GME. It is therefore necessary that sufficient number of *high-confidence inliers* that cover a majority of the image content are present at the end of the *2-RanSaC* step, for the GME to be declared successful.

Finally, the online evaluation of the GME is performed as follows:

- The GME is considered successful as in Fig. 4 (a) if there are sufficient number of *high-confidence inliers* satisfying the spatial distribution constraint, with low location errors.

- The GME is considered to have failed due to a small number of incorrect feature correspondences *skewing* the estimation as in Fig. 4 (b) when the *high-confidence inliers* have high inlier agreement but fail the spatial distribution check.

- The GME is considered to have failed as in Fig. 4 (c) when both the *high-confidence inliers* fail both the checks for inlier agreement as well as spatial distribution.

As seen in Fig. 5, with the *2-RanSaC*, two models for homography H_1 and H_2 are estimated. In the case of a successful GME, H_2 with the tighter threshold T_2 is reported if it passes the inlier agreement check, otherwise H_1 computed with threshold T_1 is reported.

Figure 3 illustrates the flow of the evaluation steps described thus far, for the GME with the sparse set. A sparse set of feature correspondences are selected in phase 1 (P1). RanSaC is applied with two thresholds T_1 and T_2 as shown in Fig. 5 in the step: *2-RanSaC*, providing the estimated homographies H_1 and H_2 respectively and the *high-confidence inliers*. Next, the inlier agreement is checked for the *high-confidence inliers* with H_1 and H_2. It is also checked if sufficient *high-confidence inliers* meet the spatial distribution constraint. If both these checks are passed, the model among H_1 and H_2 that had better inlier agreement, is reported. When either of these checks fails, a systematic repopulation as shown in phases P2-P4 is undertaken, as described in the next subsection.

4.2. Repopulation

Repopulation with additional features, to improve the accuracy of the GME, is performed in a phased manner, when a failure is detected by the proposed online evaluation of GME strategy. Phase 1 (P1) represents the first pass of the estimation with a sparse well-distributed feature set with n_1 features per block in a $k * k$ grid. When the estimation in P1 fails, subsequent phases of repopulation are performed, as detailed below:

1. *Selective injection of alternate features*: When the feature correspondences are lost during the *2-RanSaC* step - as either outliers or *shaky* features, the spatial distribution check fails, as there are not sufficient well-spread *high-confidence* inliers. This GME failure scenario corresponds to Fig. 4 (b). A selective injection of features only in the blocks that have lost representation is performed, referred to as phase P2 in Fig. 3. It is to be noted that the spatial distribution check is performed after the check for inlier agreement. Therefore, the *high-confidence inliers* are in high degree of agreement. Additional features are needed only to *support* the homography model estimated by the current GME. Therefore, in the blocks that have lost representation, additional feature correspondences are generated. For each such alternate feature, it is determined if it *supports* the current estimation by checking if its location error, using Eq. 3, is small. If sufficient *supporters* with these alternate features are found, the corresponding homography H is reported. This process is undertaken for H_2 first, if it passes the inlier agreement check and if not, then H_1 is considered.

2. *Uniform increase in density of features*: If neither of the estimates H_1 and H_2 pass the inlier agreement check (i.e. have low location errors for all their high-confidence inliers), then it indicates that the confidence in the estimation by the sparse set in P1 is low. This corresponds to the GME failure scenario in Fig. 4 (c). A higher density of features, n_2 ($> n_1$), is then applied and the inlier agreement is determined as shown in phase P3. If this fails as well, then the estimation is performed with the worst case dense set of features n_3 ($> n_2$) in phase P4.

4.3. Robust Tracking

In the presence of drastic camera motions, such as those experienced during bank turns by aerial vehicles, the accuracy of the KLT feature tracking falls [23] leading to an increase in the noisy feature correspondences used for GME.

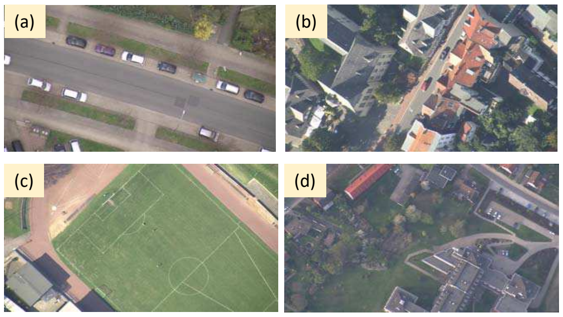


Figure 6. Aerial video datasets [16] for evaluation (a) *TNT-350m* (b) *TNT-500m* (c) *TNT-1000m* (d) *TNT-1500m*

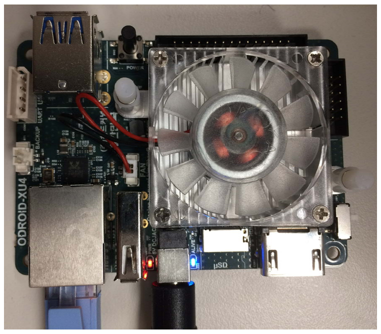


Figure 7. Odroid XU-4 platform [3]

In this case, increasing the density of features does not guarantee improved GME accuracy as the additional features employed are also likely to be noisy. Instead, a robust feature tracker such as [20] can be employed with the sparse feature sets in the proposed sparse-GME method.

5. Performance Evaluations

The proposed sparse-GME method is evaluated against a conventional approach that typically uses a dense set of features, irrespective of scene conditions. This approach, referred to as dense-GME in this paper, uses 1000 Shi-Tomasi corners [22] as features. In the sparse-GME, the Shi-Tomasi corners are selected in a block-based manner in a 4x4 grid ($k = 4$), with these number of features detected per block in each phase of repopulation: $n_1 = 1$, $n_2 = 5$ and $n_3 = 10$. The KLT tracker [14] is applied to compute the feature correspondences. RanSaC [9] algorithm is used with a reprojection threshold $T = 3.0$ for the dense-GME. In contrast, the sparse-GME applies RanSaC with two thresholds $T_2 = 3.0$ and $T_1 = 1.0$. Additionally, for the inlier agreement criterion, the threshold on the location error, $T_{LE} = 0.5$ is used. The check on spatial distribution requires *all high-confidence inliers* in phase P1 and 80% of the *high-confidence inliers* in phase P3 to have a low location error.

Experimental Platforms: Similar to the plethora of existing work, we evaluated the proposed sparse-GME technique first on a laptop PC (specifications - Intel Core i7-2670QM CPU @ 2.20GHz and 8 GB RAM, running Windows 7). In addition, we also evaluated the proposed techniques on an ARM processor based mobile embedded development platform, Odroid XU4 [3] as shown in Fig. 7. This allowed us to validate the effectiveness of the sparse-GME method on a platform with comparable processing capabilities as on a UAV. The XU4 platform runs on a Ubuntu 14.04 LTS Linux OS and contains an octa-core Samsung Exynos 5422 with 4 high performance/high power Cortex A15 cores and 4 low performance/low power Cortex A7 cores in a big.LITTLE arrangement. We pinned the sparse-GME to a single A15 core for our experiments to simulate an extremely constrained UAV platform.

Aerial Video Dataset: We have used the video sequences from the *TNT Aerial* dataset [16] for our experiments, as shown in Fig. 6. These videos were recorded on a UAV with a downward looking camera with a full-HD resolution (1920x1080, 30 fps). To evaluate the accuracy of GME performed on a pair of frames I and J, the background peak signal-to-noise ratio (PSNR), as presented in [11], is calculated between the actual frame I and the estimated frame $I' = H(J)$, where H is the homography reported by the GME.

The moving objects (foreground) are excluded from the PSNR computation so that the differences are related to only GME errors and non-foreground changes between the frames. The PSNR is computed as:

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{5}$$

Here, MSE is the mean of the squared-error of the pixel intensities between the actual frame I and estimated frame I' after the foreground has been removed.

For the video dataset, we obtain the absolute difference in the background PSNR between the outputs from the sparse-GME and the conventional dense-GME methods. This absolute difference is represented as $\Delta\mathrm{PSNR}$ and computed as follows:

$$\Delta\mathrm{PSNR} = |\mathrm{PSNR}_{\mathrm{sparse}} - \mathrm{PSNR}_{\mathrm{dense}}| \tag{6}$$

We also count the number of frames that have a PSNR difference less than a pre-defined threshold value (T_{PSNR}), and report them as successful cases of GME. We have used ($T_{\mathrm{PSNR}} = 1$ dB) for our evaluations. The number of successful frames (ϕ) given total number of n_F frames, can be calculated as follows:

$$\phi_i = \begin{cases} 1 & \Delta\mathrm{PSNR}_i < T_{\mathrm{PSNR}} \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

$$\phi = \frac{\sum_{i=1}^{n_F} \phi_i}{n_F} \tag{8}$$

Table 1. Accuracy results of sparse-GME with aerial video datasets [16]

Dataset	Avg. ΔPSNR (dB)	Successful frames ϕ (%)
350m	-0.07	100
750m	-0.20	97.5
1000m	-0.14	99.7
1500m	-0.08	99.7

Table 2. Efficiency results of sparse-GME for aerial video datasets [16]

	Speedup (%)		Repopulation Phases (%)			
	ψ_{PC}	ψ_{Embedded}	P1	P2	P3	P4
350m	46.8	39.6	97.4	2.6	0	0
750m	44.4	38.8	94.9	2.6	2.6	0
1000m	43.1	37.3	98.0	0.4	1.5	0
1500m	42.7	44.2	98.7	0.1	1.1	0

Table 1 shows the accuracy results for the video sequences. The sparse-GME achieves background PSNR similar (i.e. within 1 dB) to the baseline dense-GME for over 97% of the frames in all the video sequences. This is represented as the % of successful frames, ϕ, in the second column of Table 1. Column 1 shows that the average difference in background PSNR (ΔPSNR) while using the proposed sparse-GME method in place of the conventional dense-GME, is less than -0.2 dB. This clearly proves that the sparse-GME method achieves comparable accuracy as dense-GME.

The average execution time per frame for all video sequences is 149 ms for dense-GME. On the other hand, the average execution time drops to 83 ms when the proposed sparse-GME is applied. In addition, we also measured the execution time on the Odroid-XU4 platform [3]. On the Odroid-XU4 platform, the average execution time per frame for all video sequences is 679 ms for dense-GME and drops to 408 ms for sparse-GME.

We computed the relative speedup in execution time (ψ) achieved with sparse-GME in comparison to dense-GME as:

$$\psi = \frac{(t_{\text{dense}} - t_{\text{sparse}})}{t_{\text{dense}}} \tag{9}$$

The relative speedup is shown in columns 1 and 2 of Table 2. While, expectedly, the average execution time on the Odroid-XU4 platform increases significantly for both sparse- and dense- GME methods, the relative speedup remains similar to the one achieved on the laptop PC. This speedup can be directly attributed to the substantial reduction in the number of feature correspondences that need to be computed for sparse-GME that eventually leads to computations saved in the feature tracking stage. With the sparse-GME for each frame, the GME is evaluated by employing the proposed *2-RanSaC* strategy, as described in Section 4.1. However, the savings achieved by remaining sparse far outweighs this additional computation.

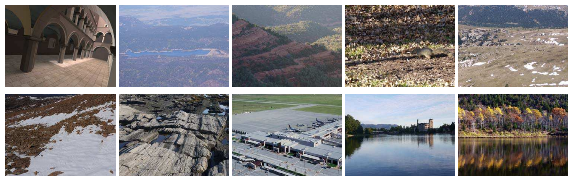


Figure 8. Images used for the simulated data for evaluation of the proposed sparse-GME method

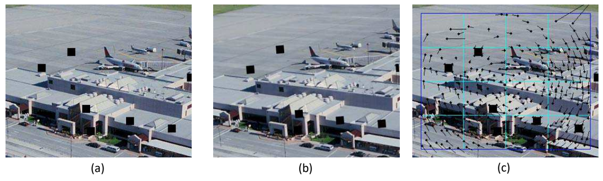
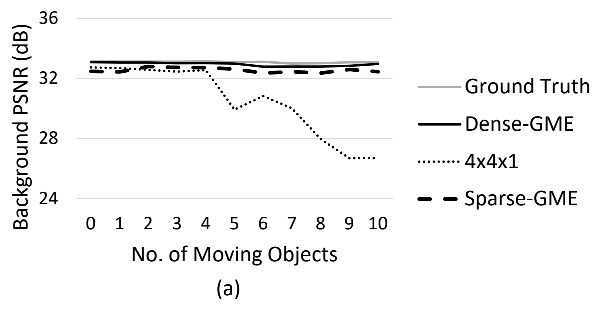


Figure 9. Simulated Frames: (a) Frame1 (b) Frame 2 (c) Optical flow generated by the KLT tracker on Frame 1; 7 moving objects with in-plane rotation of 5 degrees

In Table 2, the repopulation phases needed for the sparse-GME, are also shown for each sequence in Columns 4 through 7. These results were nearly identical for the PC and the Odroid-XU4, as expected, since they are independent of the platforms used. It can be observed that in over 94% of the frames, for all the video sequences, sparse-GME is able to complete with only a single phase P1. The slightly denser phases P3 and P4 are only needed by at most 2.6% of the frames. This shows that the proposed method is effective in detecting when the GME has been successful, leading to substantial savings in computations by using a very sparse set of features for most of the aerial video frames.

Simulated frames: As the videos from the TNT dataset do not contain challenging scenarios with significant camera motion and large number of moving objects, we generated simulated data to evaluate the proposed method for such scenarios. We use the images shown in Fig. 8 to generate simulated frames for these evaluations. In Fig. 9, the applied simulations are illustrated. Moving objects are simulated using overlaid black square patches. Random local motion is applied to the simulated objects in a range of ± 10 pixels. The simulated frames contain 0-10 moving objects. The camera motion is simulated as 1-10 degrees of in-plane rotation. All the frames have a resolution of 352x288.

The proposed sparse-GME is also compared with a naive approach of only using sparse features in the phase P1, without employing the repopulation step. This naive approach is referred to as 4x4x1 for our discussions. Figure 10 (a) shows the evaluation results for simulated frames with 0-10 moving objects in the frames (with a camera motion of in-plane rotation of 2 degrees). As seen in the figure, the



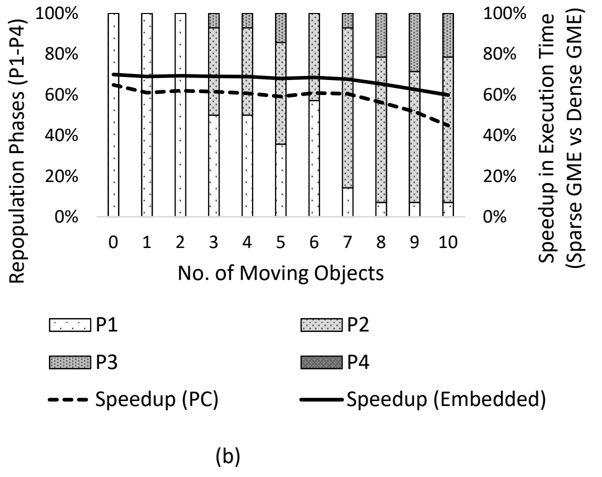


Figure 10. Evaluation results of sparse-GME with complex scene condition (large number of moving objects) (a) Accuracy (b) Efficiency

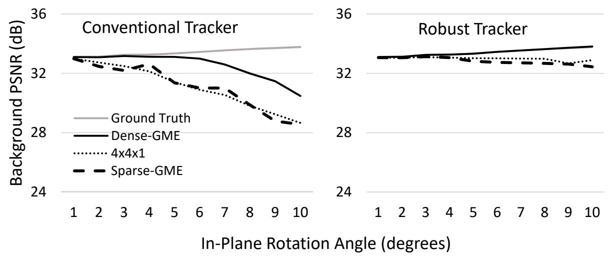


Figure 11. Accuracy of sparse-GME with a conventional KLT tracker [14] and robust KLT tracker [20] for drastic camera motion (simulated in-plane rotation)

sparse-GME matches the accuracy of the baseline dense-GME, even with a large number of moving objects. Clearly, as the number of moving objects increases, the naive sparse feature set in 4x4x1 fails, while the sparse-GME successfully detects when the sparse feature set in Phase P1 fails, and ramps up the required number of features, thereby achieving good GME accuracy. Figure 10 (b) shows that as number of moving objects increases, the sparse-GME employs the later phases of repopulation P2-P4 to increase the feature correspondences representing global motion. However, it is noteworthy that an average speedup in execution time of over 45% on PC and 60% on the embedded Odroid platform, is still achieved by the sparse-GME when compared to the dense-GME.

Figure 11 (a) shows the results when an in-plane rotation of 1-10 degrees is applied. It can be observed that in this case the accuracy of GME falls for both the sparse as well as dense feature sets, due to complex inter-frame camera motion. This is because the conventional KLT feature tracker is unable to deal with the drastic distortions caused by the rotation, resulting in inaccurate feature correspondences. Therefore, the proposed method also deteriorates in performance as it is using a subset of the KLT feature correspondences used by the dense-GME. However by employing a robust feature tracker such as the one presented in [20], the accuracy of GME can be substantially improved for both the dense-GME and sparse-GME, as shown in Fig. 11 (b). With the robust feature tracker, the sparse-GME still leads to 37-69% speedup in execution time on PC, compared to the dense-GME.

6. Conclusion

In this paper, we presented a strategy for low-complexity global motion estimation technique with minimal number of features, that adapts the number of features needed for GME with the complexity of the scene and the camera motion. This leads to substantial reduction in computations for GME in aerial surveillance videos. It was shown that by ensuring good spatial distribution, a very sparse set of features is sufficient for accurate GME in typical aerial surveillance videos which have predominantly simple scene conditions. In addition, a computationally simple online evaluation strategy for GME was proposed that can rapidly detect failure in GME when scene conditions become complex. A systematic repopulation process is employed in such cases when failure is detected. Our extensive evaluations show a substantial speedup in execution time of around 40% without compromising accuracy on both a laptop PC as well as a mobile embedded platform that closely resembles the onboard processing capabilities of a UAV. We also show that for complex scene conditions due to significant camera rotations, high accuracy of GME can be achieved with very sparse set of features when a robust feature tracking is employed. The proposed sparse-GME method therefore enables low-complexity GME for rapid processing of aerial surveillance videos. In future, optimized implementations of the proposed method will be explored that efficiently exploit the multi-core CPU and the on-chip GPU in the Odroid platform. In addition, we would also like to study and optimize the energy consumption of the proposed techniques.

References

[1] Conservation drones. http://conservationdrones.org/. Accessed: 2017-03-16. 1

[2] DJI Manifold. https://goo.gl/k08obE. Accessed: 2017-03-16. 2

[3] Odroid-XU4. http://goo.gl/Nn6z3O. Accessed: 2017-03-16. 2, 6, 7

[4] RanSaC in 2011 (30 years after). http://www.imgfsr.com/CVPR2011/Tutorial6/RANSAC_CVPR2011.pdf. Accessed: 2017-03-16. 2

[5] E. Bostanci. Enhanced image feature coverage: Keypoint selection using genetic algorithms. *arXiv preprint arXiv:1512.03155*, 2015. 3

[6] O. Chum and J. Matas. Randomized RANSAC with Td, d test. In *British Machine Vision Conference (BMVC)*, volume 2, pages 448–457, 2002. 2

[7] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 220–226. IEEE, 2005. 2

[8] S. Ehsan, A. F. Clark, and K. D. McDonald-Maier. Rapid online analysis of local feature detectors and their complementarity. *Sensors*, 13:10876–10907, 2013. 3

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2, 6

[10] O. Gallo, A. Troccoli, J. Hu, K. Pulli, and J. Kautz. Locally non-rigid registration for mobile HDR photography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 49–56, 2015. 2

[11] M. N. Haque, M. Biswas, M. R. Pickering, and M. R. Frater. A low-complexity image registration algorithm for global motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(3):426–433, 2012. 6

[12] D. Hulens, T. Goedemé, and J. Verbeke. How to choose the best embedded processing platform for on-board UAV image processing? In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 1–10. IEEE, 2015. 1

[13] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, et al. Aerial video surveillance and exploitation. *Proceedings of the IEEE*, 89(10):1518–1539, 2001. 3

[14] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. 2, 6, 8

[15] P. Mainali, Q. Yang, G. Lafruit, L. Van Gool, and R. Lauwereins. Robust low complexity corner detector. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):435–445, 2011. 2

[16] H. Meuel, M. Munderloh, M. Reso, and J. Ostermann. Mesh-based piecewise planar motion compensation and optical flow clustering for ROI coding. *APSIPA Transactions on Signal and Information Processing*, 4, 2015. 1, 6, 7

[17] A. H. Nguyen, M. R. Pickering, and A. Lambert. The FPGA implementation of a one-bit-per-pixel image registration algorithm. *Journal of Real-Time Image Processing*, 11(4):799–815, 2016. 2

[18] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005. 2

[19] J. Paul, W. Stechele, M. Kröhnert, T. Asfour, B. Oechslein, C. Erhardt, J. Schedel, D. Lohmann, and W. Schröder-Preikschat. Resource-aware Harris corner detection based on adaptive pruning. In *International Conference on Architecture of Computing Systems*, pages 1–12. Springer, 2014. 2

[20] N. Ramakrishnan, T. Srikanthan, S. K. Lam, and G. R. Tulsulkar. Adaptive window strategy for high-speed and robust KLT feature tracker. In *Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, pages 355–367. Springer, 2015. 6, 8

[21] S. Shah. Real-time image processing on low cost embedded computers. *Techincal report No. UCB/EECS-2014–117*, 2014. 2

[22] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600. IEEE, 1994. 1, 2, 6

[23] S. Tanathong and I. Lee. Translation-based KLT tracker under severe camera rotation using GPS/INS data. *IEEE Geoscience and Remote Sensing Letters*, 11(1):64–68, 2014. 5

[24] M. Teutsch and W. Krüger. Detection, segmentation, and tracking of moving objects in UAV videos. In *Ninth International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 313–318. IEEE, 2012. 1, 2

[25] K. Whitehead and C. H. Hugenholtz. Remote sensing of the environment with small unmanned aircraft systems (UASs), part 1: A review of progress and challenges. *Journal of Unmanned Vehicle Systems*, 2(3):69–85, 2014. 1, 3