Efficient Video Stabilization via Partial Block Phase Correlation on Edge GPUs

Cevahir Çığla
Aselsan Inc.
Ankara, Turkey
cciglaQaselsan.com.tr

Abstract

In this paper, an efficient video stabilization method is introduced that exploits phase correlation on partial blocks. The approach addresses very low computational complexity on edge GPU devices for surveillance cameras. The global motion of consecutive frames due to camera vibrations along horizontal and vertical axes are extracted through phase correlation of informative blocks among the video. The novelty of the proposed approach lies within the extraction of sub-blocks with high suitability/reliability and reliable merge of multiple block correlation maps. Pipeline implementation of detection of informative blocks and utilization of 4-8 sub-blocks during the shift estimation are the key features of enabling up to 150 fps processing for HD (1920x1080) video on Nvidia Jetson nano edge GPU device without any accuracy loss. Utilizing a generic scheme and reasoning as well as efficient GPU implementations, proposed approach is highly adaptable to various video content by altering sub-block choices. Considering stabilization as a pre-process step, the proposed algorithm reserves sufficient computational room for further video analyses on edge compute devices.

1. Introduction

The advances in computational capacity of edge devices enabled excessive use of cameras and algorithms for a large variety of applications. Motion detection [34], smart object detection-tracking [2],[5] and attribute classification for human/vehicle re-identification [31] are the most endeavoured features applied on outdoor cameras for city, facility and border surveillance. The common assumption behind these surveillance algorithms is the immobility of the cameras that enables use of consistent temporal data processing. In this way, it is possible to interpret the changes in a scene for motion detection, temporally associate frame-wise detected objects and provide multi-object tracking. Besides, the details in the scenes are observed consistently and not affected from motion blur.

On the other hand, outdoor is an uncontrolled environment and the weather conditions affect the use of algorithms in different manner. One of the most important effects is the violation of the immobility of the cameras due to wind, pole or basement vibrations. Little amount of vibration along the camera mount is magnified by the camera field-of-view (FOV). These vibrations mostly produce horizontal and vertical shifts between consecutive frames that decrease the sharpness, spatial resolution as well as introduce many relative artificial movements in the videos. Besides, the shaky videos introduce eye fatigues and undesired observation experience for the security staff.

Though, video stabilization is an important task for outdoor computer vision applications. The are two main approaches to handle this problem, optical image stabilization (OIS) and digital image stabilization (DIS). OIS is achieved by additional hardware installment on the cameras, such as inertial measurement units, that transfers the metric shift on the camera head to the camera software that reverses the motion according to its FOV. On the other hand, DIS does not require additional hardware where image processing techniques are utilized to estimate the shifts among the consecutive frames and stabilize the video frames. OIS solutions do not depend on the image content and are more expensive depending on the sensitivity of IMUs; whereas DIS provides cost efficient and light-weight solution that is limited by the image content. This limitation is not valid as long as the observed scenes involve sufficient features, texture or content. That assumption is mostly met under surveillance scenarios where streets, facility borders and city squares involve various type of installations around.

At that point, it is important to note that the main purpose of image stabilization is providing stable video frames for computer vision applications. DIS is considered as a preprocess step that is supposed to enhance image quality and temporal consistency. Though, the computational complexity of video stabilization should be as low as possible that leaves computational room for the main algorithms. This efficiency is vital for new age computer vision algorithms that are based on deep neural networks (DNNs) executed mostly on mobile edge GPUs, such as Nvidia Jetson family.

This paper introduces an efficient video stabilization technique that requires low GPU utilization without losing accuracy. Piece-wise phase correlation and block decision are the key features of the proposed algorithm that are totally implemented on GPU such that up to 150 fps stabilization is enabled on Jetson nano [1]. Giving the related work on video stabilization, the details of the proposed algorithm are given in the third section. In the following section, the efficiency and accuracy of the proposed method are shown by detailed experiments. The paper concludes by further discussions and future directions in the final section.

2. Related Work

Video stabilization has been popular recently for online-offline video editing, smooth video capture through a moving camera in addition to wide use case of pre-processing step for computer vision application. The main blocks of video stabilization [11] is the estimation of camera movement among consecutive frames, motion modeling to filter out spikes in the movement and finally warping (rendering) of current frame into its stabilized version. In traditional computer vision methods, each block is treated independently while the recent neural network based techniques [8],[30],[19],[32] merge these steps into same pipeline.

2.1. Scope of this Study

This paper considers the stabilization problem as a pre-processing step rather than the main purpose of providing visually pleasing videos from shaky moving camera captures. Moreover, it is assumed that videos are captured through stationary cameras where the vibrations due to wind, pole or camera mount are the cause global horizontal and vertical frame shifts among consecutive frames. The output of stabilization, a.k.a temporally stable videos, can be processed further on the edge for various computer vision applications such as motion estimation [34], smart object detection [2] and multi-object tracking [5]. In this scenario, additional algorithms are considered on the edge, therefore limited computational complexity is required for video stabilization. Besides, zero motion models are forced for the surveillance cameras because of their stationary movement characteristics. Throughout this study, edge processing platform is considered to be NVIDIA Jetson nano, that enable prompt processing with mobile GPUs installable into cameras.

A comprehensive review of DIS is given in [11], on the other hand we classify DIS techniques into three groups according to the utilization of representations for motion estimation and rendering. The first group of algorithms rely on the corner or feature points within the scene that have high saliency to be tracked. The second group exploits phase correlation that enables use of large dense pixel groups efficiently to extract global shifts in Fourier domain. Finally, the recent advances in neural networks have enabled deep learning techniques as in most computer vision application, as an alternative video stabilization family.

2.2. Feature based Stabilization

In this category, corners points or feature points are exploited to get representation of a frame and several matching techniques are utilized to associate those features among consecutive frames. SIFT[20] FAST [26], SURF [4] and ORB [27] are the most endeavored feature representations in computer vision.

[14] and [3] exploit the fundamental frame work of feature based methods, including feature detection/tracking and motion modelling to smooth-out un-desired vibrations. [21] utilizes FAST [26] feature detector and then BRIEF [7] representation to match those features among consecutive frames. The paper optimizes for mobile GPU and runs on Jetson nano and achieves 81 fps for HD videos. Similarly, [13] performs stabilization in 30 ms on GTX2080 GPU for 720p videos by use of SURF features on GPU. [33] incorporates feature based tracking and piece-wise scene planarity modeling to warp consecutive frames in a content preserving manner. [17] utilizes horizontal and vertical gradients to detect corner points and then represents each feature point within *11x11* blocks. After a brute force search, the best feature matches provide the temporal motion shifts. The frame shifts are smoothed temporally to handle shaky videos that is common for hand held cameras. The paper compares well known feature representations FAST [26], SURF [4] and ORB [27], where they improve execution speed by a factor of 2 compared to FAST.

2.3. Phase Correlation based Stabilization

Phase correlation (PC) is a fundamental tool that provides efficient convolutions in time domain as a single step multiplication in frequency (Fourier) domain [28]. It has been excessively used for single object tracking [6],[12],[9] optical flow estimation [25] due to fast and prompt execution. Therefore, phase correlation has been a strong candidate for real-time stabilization methods.

[24], [16] and [15] utilize PC on full frames as a single large block for the estimation of frame-wise shift. The methods provide high stabilization quality as long as the image contents involve sufficient texture as a whole distribution. [22] analyses several sub-blocks that cover various parts of the image with alternating sizes then phase correlation is conducted on the blocks to determine frame rotation angle. The global shifts are estimated via cross correlation where whole image is exploited as a single large block for phase correlation. Utilizing different approaches for angle and translational motion introduces computational complexity for this study. Four blocks with fixed size and

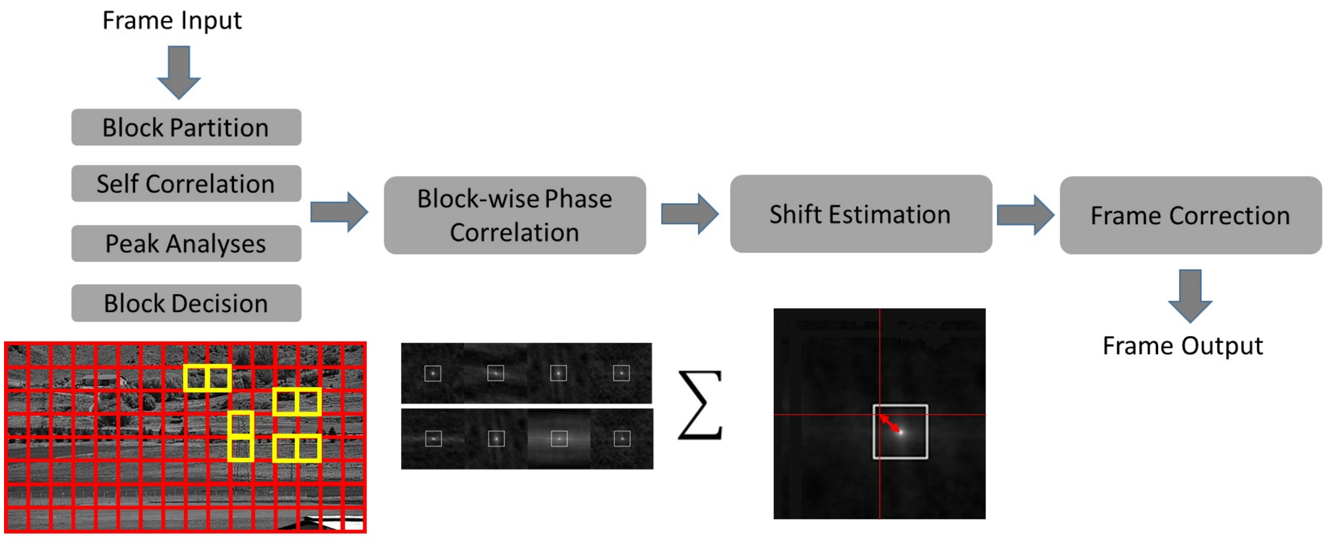

Figure 1. The flowchart of the proposed approach.

location are utilized in [10] in order to estimate pixel shifts via phase correlation. Use of fixed blocks prevent this approach to be adaptive to various image content as well as spatial shift in time.

2.4. Neural Network based Stabilization

These approaches mostly focus on video editing purpose where stabilization is considered to the final outcome of the processing. Therefore, they do not constrain computational efficiency as the main concern and tackle the problem as providing the most visually pleasing and stable videos. [8] consists of three sub-networks, coarse-fine-margin nets, which consecutively include transformation estimator, scene parallax reduction and an in-painting module. These techniques provide higher quality stability among so called real-time neural network based techniques. [19] gets multiple frames as input to their unified neural network and outputs rendered frames in order to provide visually pleasing video capture from a hand-held camera. It requires high computational power that is far from being a cost efficient pre-processing step. StabNet [30] focuses on transformation estimation approach through convolutional neural networks where homography models are estimated per frame and stable camera path is created for moving camera captures. It is an efficient online alternative to offline stabilizers with up to x30 speed-up on desktop GPUs such as NVIDIA GTX 1080 Ti. [32] exploits neural networks to estimate pixel-wise optical flow among consecutive frames. The flow information is utilized to warp frames via motion in-painting and smooth rendering.

According to the scope of this study, all of the neural network based approaches, including real-time performers as well, are computationally demanding and exploits desktop GPUs to target 25 fps videos. This is not practical and applicable for edge-devices where mobile GPUs such as Jetson family (tx1, tx2, nano) are under consideration. Besides, learning-based methods highly depend on the training data and can suffer from large motions. Due to the lack of comprehensive training data sets that cover all cases in surveillance, the conventional methods are more robust and perform better in a general setting. In addition to the techniques listed previously there are efficient alternatives that exploits motion vectors extracted during video coding for stabilization. [18] performs 50 fps stabilization for 720p videos on CPU which looks promising in terms of real time performance. On the other hand, it requires a percentage of 50% for the number of valid inter motion sub-blocks in coding step, that introduces additional computational cost.

3. Proposed Approach

The proposed method mostly relies on phase correlation due to its efficiency of Fourier domain implementations (a.k.a FFT) in many platforms as an alternative to image-domain convolution. PC is utilized in the proper block determination step to identify the best blocks to track as well as in the estimation of the shifts of the blocks among consecutive frames. In general, PC is applied for pre-determined blocks whose shift detection capability is limited by the size of the blocks. Moreover, the size of PC blocks has an effect on the FFT calculation as well. We adapt traditional full frame and fixed block size approaches into sequential and multi-scale process in order to meet real-time execution and large pixel vibration handling capability.

There are four main functional blocks of the proposed method whose flowchart is given in Figure 1. First, the input image is divided in sub-blocks to determine the regions proper for phase correlation. Then, PC between patches at the same regions along consecutive frames is exploited to estimate shifts such that correlation maps of different patches are merged according to their reliability scores. Estimating the shift in the third step, stabilized images are rendered in the final step by shifting the current frame according to horizontal and vertical motion.

3.1. Sub-Block Decision

One of the most crucial step of the proposed approach is the determination of proper regions for PC within reference images. In that manner, the reference image is tiled into square blocks and each block is auto-correlated after edge images are extracted to get the characteristics and extract local feature distributions as shown in Figure 2. A block is proper for phase correlation as long as the auto-correlation maps has sharp peak at the center and low values along the outskirts. This enables clear differentiation of the block among various possible shifts such that the shift during undesired camera motion can be estimated with high reliability. It is important to note that, edge detection dims the effect of flattened and un-textured regions on the correlation maps as well as noise and highlights intensity transition regions. Though, salient pixels have much more impact on ACMs and the peak analyses provide more reliable blocks.

The reliability of the blocks are determined by comparing the center peak (R^c_{max}) and out-of-center peak energy values (R^{oc}_{max}) as well as the minimum value (R_{min}) in the auto-correlation map (ACM) according to $\rho = \frac{\sqrt{(R^c\_max - R\_min)(R^c\_max - R^{oc}\_max)}}{R^c\_max}$. The equation states that the reliability of a patch is high as long as the peak at the center is significantly higher than off-center peak and the minimum value. If these values approach to center peak, reliability decreases. The center area is considered to be a bounded region around the center covering %4 area of a patch as shown as red squares in Figure 2. The out-of-center peak is the peak value within the remaining area. Some typical patch ACMs are shown in Figure 2 where the maps in the first row have sharp peakness and high reliability while the maps in the second row belong to patches with repetitive structures that generate multiple peaks distributed around (shown in green arrows). Though, the second row maps relate to un-reliable patches that are not proper for PC. The ACMs are given for various patches in Figure 3 where proper and desired maps are circled in yellow. It is clear that some patches with consistent vertical and horizontal edges, such as the patches in the first row, have multiple peaks along the edge directions. Besides, texture free patches have flattened ACMs with insignificant center peaks. These kinds of ACM structures indicate that those patches do not provide discriminant and reliable correlation distributions. On the other hand, patches with sharp ACMs are sensitive to shifts in each direction that make them ideal for PC based shift detection. In order to provide a good balance between accuracy and computational complexity, the

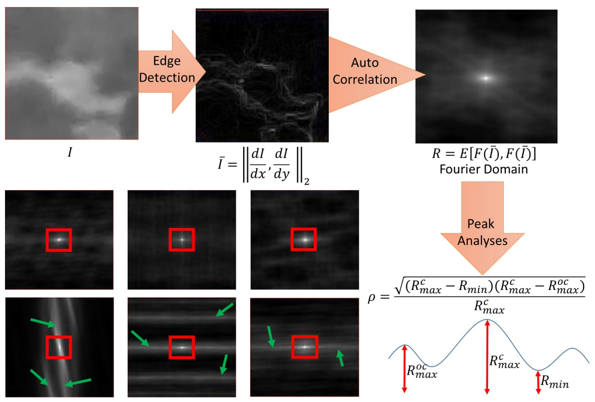
Edge Detection
Auto Correlation
I
$\bar{I} = \left\|\frac{dI}{dx}, \frac{dI}{dy}\right\|_2$
$R = E[F(\bar{I}), F(\bar{I})]$
Fourier Domain
Peak Analyses
$\rho = \frac{\sqrt{(R^c_{max} - R_{min})(R^c_{max} - R^{oc}_{max})}}{R^c_{max}}$
R^{oc}_{max}
R^c_{max}
R_{min}

Figure 2. The reliability of a block is determined via auto-correlation of edge images and center-surround peak analyses.

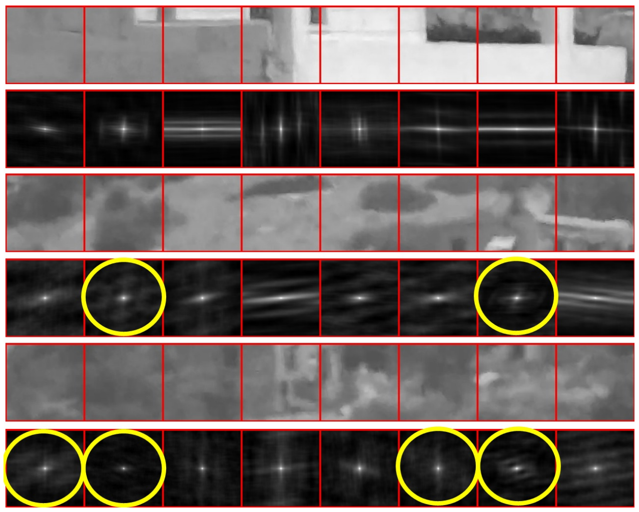

Figure 3. Some exemplary patches with their ACMs; repetitive and texture free patches have un-reliable peaks while patches shown in yellow circles have reliable ACMs.

best 4 or 8 blocks are chosen for the further steps along the reference frame; that have the top-(4,8) reliability among all sub-blocks.

3.2. Phase Correlation based Shift Estimation

In this step, pre-determined reference frame blocks (N) are utilized to compare image content in the following frame. The correlation between the blocks in the reference image (I_0) and current image (I_t) are conducted after edge detection and Fourier Transform as shown in Figure 4. For each block, reliability (ρ), horizontal shift (Δx) and vertical shift (Δy) values are extracted from the correlation map. Reliability is calculated via center-surround peakness analyses around the highest correlation value, where the location of this value also provides the shifts (Δx,Δy) of the corresponding block along consecutive frames. The final horizontal and vertical shifts between frames is achieved by the normalized weighted summation of the shift estimate of

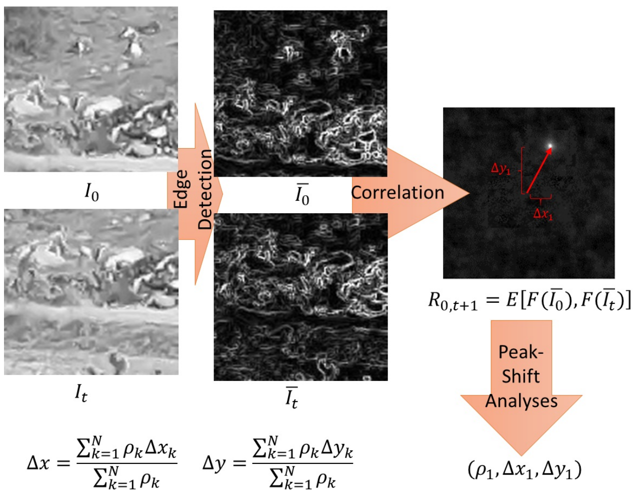
I_0
Edge Detection
$\bar{I}_0$
Correlation
Δy_1
Δx_1
$R_{0,t+1} = E[F(\bar{I}_0), F(\bar{I}_t)]$
I_t
$\bar{I}_t$
Peak-Shift Analyses
$\Delta x = \frac{\sum_{k=1}^{N} \rho_k \Delta x_k}{\sum_{k=1}^{N} \rho_k}$
$\Delta y = \frac{\sum_{k=1}^{N} \rho_k \Delta y_k}{\sum_{k=1}^{N} \rho_k}$
$(\rho_1, \Delta x_1, \Delta y_1)$

Figure 4. The flow of shift and reliability estimation between the sub-blocks of consecutive frames.

each sub-block as given in Equation 1.

$$\Delta x, y = \frac{\sum_{k=1}^{N} \rho_k \Delta x_k, y_k}{\sum_{k=1}^{N} \rho_k} \tag{1}$$

Thus, the effect of a sub-block is related by its correlation reliability at that time instant. The final correlation map has a kind of non-maxima-suppression effect where the actual peak gets more contribution. Two examples for this case are given in Figure 5; the correlation maps of the sub-blocks highlighted in the full frames are merged and much more clear correlation maps are obtained. The cluttered outskirts within sub-block correlation maps are suppressed that results in more reliable peaks and shift estimations. Once the horizontal and vertical motion vector is estimated, these shifts are applied to the current frame in reverse directions to obtain the stabilized frame with respect to the reference frame. As mentioned previously, we do not need motion modeling in this study due to the target surveillance cameras that are fixed in position and no motion is expected. Therefore, zero average motion is assumed in these kind of cameras.

3.3. Multi-scale Adaptation

One of the disadvantages of sub-block usage is the limitation of maximum shift amounts by the pre-determined size of the blocks such that *MxM* blocks can handle only M/2 pixel shift in any direction. In order to extend and overcome this limitation we adapt the proposed approach to run in multi-scale while keeping the computational complexity as low as possible. As shown in Figure 6, down-sampling the image by a factor of α and preserving the sub-block size *MxM*, the shift limitation is also increased by a factor of α. Once shift values are determined in lower resolution, they can be transferred to the actual resolution and the correlation maps can be refined over the shifted sub-blocks. In

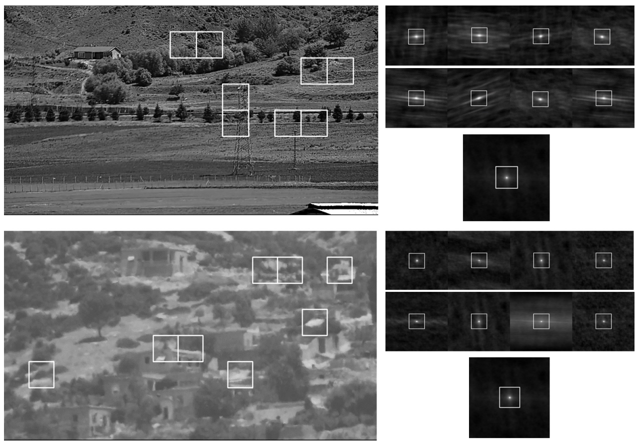

Figure 5. Merge of sub-blocks.

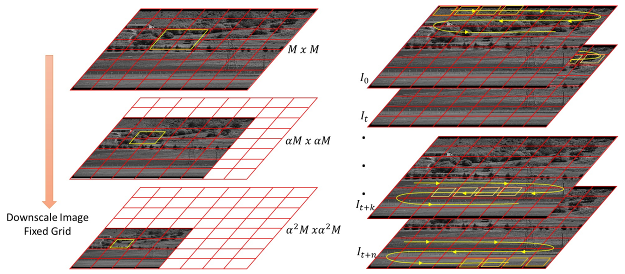

Figure 6. Left: Multi-scale extension enables larger horizontal and vertical shifts to be handled, Right: sequential proper block search, at each time a group ob blocks are checked.

Figure 6, the yellow rectangle in the lowest resolution covers a larger area in the actual frame resolution that provides a range of $[-\alpha^2 M, \alpha^2 M]$ shifts. The fine tuning and little pixel shifts can be handled in the highest resolution to compensate the precision loss due to pixel resolution decrease. This approach introduces additional pixel-shift estimation which can be kept minimum by utilizing half of the blocks (N/2) determined in the first step, while has no influence on the sub-block determination. In general, using large block size such as *128x128* is mostly sufficient to compensate *64* pixel-wide shifts; however, in case of un-expected camera shakes down-sample factor of *2-3* can be used.

This multi-scale adaptation keeps *FFT* calculation on fixed size blocks in the lower resolution image. One alternative to this approach is to utilize a larger sub-block around the most reliable smaller block. This larger block, with size of *256x256*, can handle wider shifts in images. On the other hand, single large block utilization is prone to errors due to un-textured regions within. This is a natural effect; as the area increases the probability of higher saliency decreases (or peakness reliability in this study), though spliting a large block into smaller distributed sub-blocks and searching for high reliability of PC enables higher accuracy. We also include larger block implementation throughout the experiments.

3.4. Run-time Optimizations on GPU

In this section, we give further details about the GPU and runtime optimization of the proposed approach. First of all, block based operations are well suited for parallel processing, that is actually one of the main design criteria in this study. The other criteria is the efficient correlation capability and embedded functions for Fourier-domain processing, a.k.a *FFT*, on the GPUs. It takes under milliseconds on a moderate mobile GPU to perform *FFT* for a block with size *128x128* or lower. Considering the stabilization as a pre-process step before additional computer vision analyses, it is very important to make use of these computational advantages.

It is known that *FTT* is optimized for square matrices having dimensions in the powers of *2*, thus we exploit blocks of size *128x128* for the full HD (*1920x1080*) videos. We also exploit a down-sample factor of *2* which enables up to 128 pixel shifts in horizontal and vertical axes. Under these circumstances, there are roughly *15x8* blocks and we choose *8* as the upper limit for number of proper blocks. This upper bound is due to the computational complexity constraint as mentioned previously, stabilization should be as fast as possible. In order to find proper blocks, all of the blocks are required to be auto-correlated which takes significant amount of time, violating the real-time execution. This is handled by sequential processing as summarized on the right hand side of Figure 6, where at each time instant, limited number of blocks, such as *8*, is analyzed. At each frame, different blocks are processed until all the blocks are visited. Thus, the block decision is spread along *15* frames, and stabilization starts after the final group of blocks are analyzed. This kind of pipeline processing enables controlled time complexity per frame and prevents frame skips. One drawback of this approach is that stabilization is not provided for the first *15* frames that seems to be not that critical. Once a decision is done, stabilization is conducted through the chosen blocks as long as the PC scores are above a threshold. The following decision processes, if required due to scene/illumination change, can be performed at any *15* frames time window that will not interfere with stabilization. The square structure of blocks enables prompt GPU partitioning such that each block is divided into four GPU blocks that are analyzed via *16x16* thread structure as summarized in Figure 7. All the processes apart from *FFT* and inverse *FFT*, are performed by that thread-block tiling. The Fourier domain operations are conducted on *128x128* sub-blocks in group operations such that, the sub-blocks are batched then *FFT* and *iFTT* are performed in single task. This batch processing enables faster computation compared to a single larger block of same pixel area. The correlation between the instant proper sub-blocks and the reference sub-blocks is calculated by Fourier domain multiplications and then inverse Fourier Transform.

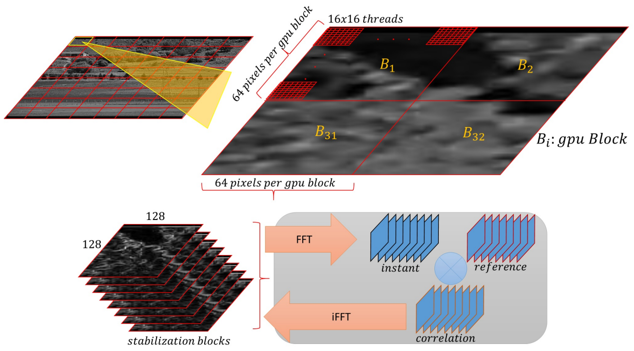

Figure 7. Two frames with their proper sub-blocks and the accumulated correlation maps.

The correlation maps are analyzed by the same GPU tiling given in the upper part of Figure 7 in order to get the peak *x,y* position and reliability ρ.

4. Experimental Results

As in most of the efficient algorithm implementations, the experimental results in this study are conducted in two phases, stabilization accuracy and computational complexity. Throughout the experiments, full HD resolution videos (*1920x1080*) are perturbed with random horizontal and vertical global shifts to model camera vibrations and their resultant views. Thus, controlled ground truth data can be utilized to measure stabilization performance numerically rather than visual interpretation. This type of perturbation is valid for surveillance cameras where mostly horizontal and vertical motion is observed rather than any rotation effect. Therefore, rolling shutter effect is out of the scope of this study where special attention is required per horizontal line. The range of perturbations is chosen under uniform distribution in the range of [-120,120] at each frame in both horizontal and vertical axes. These perturbations produce highly shaky videos with high intra-frame pixel shifts that provides a challenge for stabilization. We utilize videos from AI-City Challenge 2021 *Track 4* [23] and SOMPT22 [29] that include wide view outdoor traffic and surveillance videos. [23] involves low texture traffic high-way videos while [29] involves high texture city and square views. The choice behind these datasets is based on their surveillance view points as well as texture characteristics that help the comprehension of the capability of PC and FT for stabilization. Some sample images from these datasets are shown in Figure 8. The implementation platform is chosen to be Nvidia Jetson nano [1], that is a low cost mobile GPU platform adaptable for edge operations.

4.1. Stabilization Accuracy

We compare the proposed PC based approach with [14] where a low cost (has almost similar steps with [21]) traditional feature based stabilization is applied based on widely

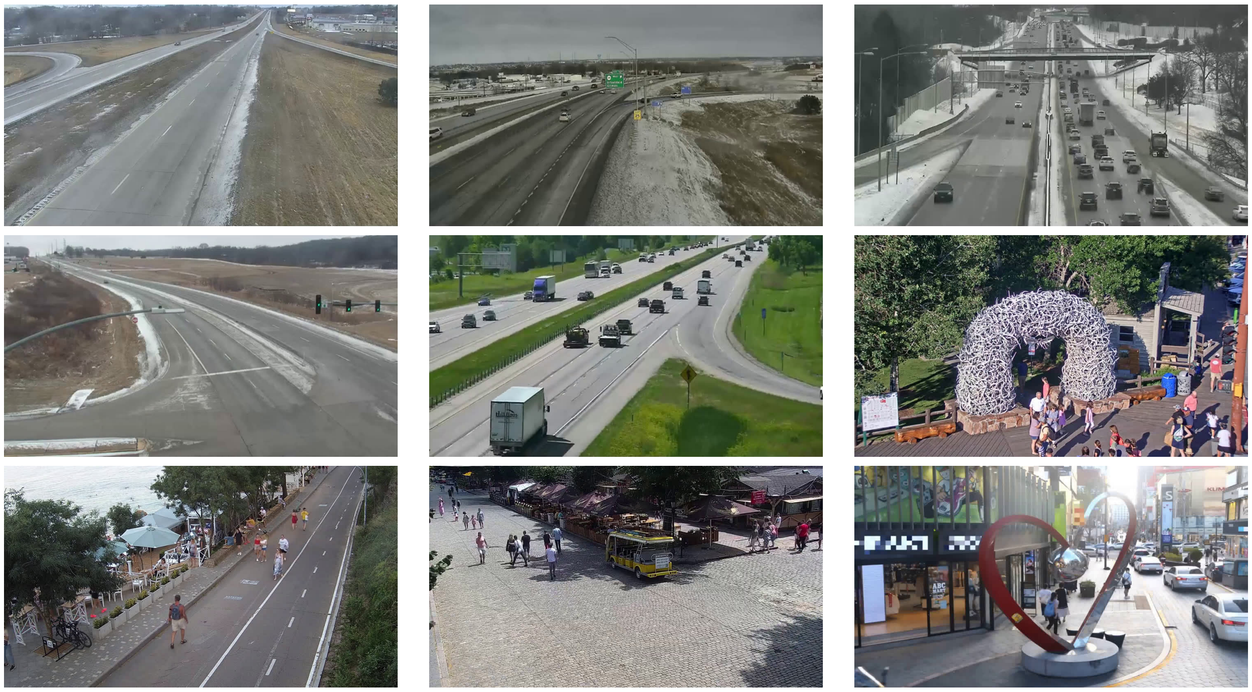
Figure 8. Samples from [23] (first row and first two images in the second row) and [29] (last row and the last image in the second row) that provide comprehensive outdoor surveillance dataset utilized for stabilization experiments in this study.

utilized feature trackers (FT) and image warping algorithms given by *openCV* library. It is important to note that these studies are adapted to horizontal-vertical shift handling by ignoring affine transformation step to decrease computation as well. These methods enable a generic comparison between phase correlation and feature based methods considering the fact that no special tricks are applied for the matching step. Both techniques rely on the generalization of mathematical techniques in phase correlation and feature tracking. For the sake of completeness, we also implement a fixed single large block based PC method that gets a larger block (*256x256*) around the best sub-block in our pipeline. The stabilization accuracy is measured by mean-squared distance between the actual and estimated shifts on various perturbed full HD videos. The results calculated over 20 videos of 1 minute-length and the random shifts are generated in five different simulations. The results are given in Table 1 with respect to datasets which indicate that proposed approach with *8* sub-block utilization gives the best stabilization accuracy in both datasets. As the texture within video increases, the performance of all techniques increases, in our case SOMPT22 [29] has more texture, therefore stabilization is more accurate on this dataset. On the other hand, PC based methods improve better as the texture and edge distribution become sharper compared to FT methods. In AI-City videos, FT and 8 sub-block PC have same stabilization accuracy, while the performance drops for lower number of sub-block utilization. It is clear that, weighted merge of ACMs of sub-blocks and the non-maxima-suppression play an important role to improve accuracy. Another important factor apart from texture characteristics is the video content of the surveillance camera, such that SOMPT22 involve larger and more abundant moving objects within the scene compared to AI-City. The slight improvement in the performance of FT compared to PC may relate to the contribution of independently moving objects within the scene, where false shifts due to large objects' motion can affect the global motion estimation of FT methods as opposed to the increase in texture. Thus, a predetermination of non-moving regions may be required to handle such false contributions. On the over all, the average MSE among all videos is around *0.3* pixel for the best performing *8* sub-block approach, corresponds to an error rate around quarter sub-pixel considering the range of perturbations. This very low error rate stabilization is quite adequate for further computer vision applications as well as observers of the videos.

Table 1. Mean-squared error (MSE) for proposed and feature based method.

±120 px	Feat.[14]	Proposed		
		1-block	4-block	8-block
aicity [23]	0.39	0.45	0.43	**0.39**
sompt [29]	0.31	0.21	0.19	**0.17**
all videos	0.35	0.33	0.31	**0.28**

Table 2. Computational complexity comparison.

Method	GPU	cores	FPS
[3]	Jetson TX1	256	60
[21]	Jetson Nano	128	81.4
1-block	Jetson Nano	128	259.7
Proposed 4-blocks	Jetson Nano	128	147.8
8-blocks	Jetson Nano	128	141.2

4.2. Computational Analyses

The run-time performance of the proposed approach is given in Table 2 with a comparison of two real-time methods whose computational times on mobile GPUs are gathered from [21]. We also provide the computational time of a single larger block based PC. It is clear that proposed approach with both 4 and 8 sub-block configuration enables much more efficient processing compared to [3] and [21]. Use of pre-selected sub-blocks tremendously decreases the computational complexity, besides efficient FFT implementations speed-up stabilization to a new level with only up to *7 ms* of requirement. Depending on the scene content, stabilization can also be achieved in less than *4 ms* by use of single larger block, with a little sacrifice of accuracy. Though, proposed approach provides a trade-off between single and 8 sub-block usage, that both yield high stabilization capability for proper surveillance camera views. The run-time of *FFT* and proposed pipeline are not linearly scaled to the the size of sub-blocks, such that when the size is doubled to *256x256*, the frame-per-second drops to (*165, 112* and *86*) for 1-4-8 block usage correspondingly. In that case, even larger shifts can be handled under *10 ms*.

The distribution of computation times with respect to main functional blocks are given in Figure 9 for 4 and 8 sub-block usage. It is clear that most of the computation is spent on peak analysis which involves merge of ACMs of sub-blocks, localization and reliability calculation of the peaks. That is the main reason of almost 2 time faster operation in single larger block which does not require any merging step. However, the merging step obviously improves the stabilization accuracy and provide a comprehensive and wide range of scene handling.

According to Table 1 and Table 2, use of 8-blocks enables *140* fps with very low (under %1) stabilization error for surveillance camera views and seems to be the best choice of the proposed approach. In case of sufficient texture, use of single large block is also an alternative with two times faster execution. Though, our technique saves enough computation for further computer vision analyses, being a good candidate for stabilization of surveillance cameras. Besides, the selection of sub-block number can be performed dynamically such that optimum performance is provided depending on the scene structure.

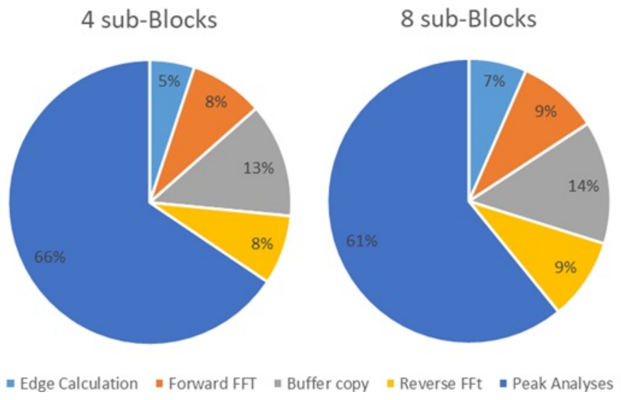

Figure 9. The distribution of execution times with respect to functional blocks.

5. Conclusion

We propose a novel and very efficient phase correlation based video stabilization on mobile GPus under non-moving and global shutter surveillance camera conditions. The method exploits determination of salient sub-block and application of block based phase correlation to this subset of blocks in order to relate recent video frames to a reference. The phase correlation is achieved by GPU optimized *FFT*, that also plays a significant role in salient block selection. The method merges the responses of each block into a final map on which the peak response provides horizontal and vertical frame shifts. Use of low number of discriminant blocks has a significant impact on the efficiency of the proposed approach. In order to handle large vibrations, a multi-scale adaptation is also provided. The experiments indicate that, stabilization on a mobile low cost GPU device, Jetson nano, takes only up to *7 ms* corresponding to roughly 150 fps without losing any stabilization accuracy. This very low cost stabilization is a good alternative for a pre-process step and saves sufficient computational space for further analyses on the camera. Balancing between accuracy and computational speed, proposed approach enables further optimizations to alternate between different number of block usages depending on the scene content. The dynamic alternation of blocks is reserved for future directions.

References

[1] Nvidia jetson nano development kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed: 2024-03-05. 2, 6

[2] Pranav Adarsh, Pratibha Rathi, and Manoj Kumar. Yolo v3-tiny: Object detection and recognition using one stage improved model. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 687–694, 2020. 1, 2

[3] S. Aldegheri, D. D. Bloisi, J. J. Blum, N. Bombieri, and A. Farinelli. Fast and power-efficient embedded software implementation of digital image stabilization for low-cost autonomous boats. In *Field and Service Robotics*, pages 129–144, Cham, 2018. Springer International Publishing. 2, 8

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia. 2

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 1, 2

[6] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010. 2

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, pages 778–792. Springer-Verlag, 2010. 2

[8] Jinsoo Choi, Jaesik Park, and In So Kweon. Self-supervised real-time video stabilization. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2021. 2, 3

[9] Cevahir Cigla, Kemal E. Sahin, and Fikret Alim. Gpu based video object tracking on ptz cameras. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 767–7678, 2018. 2

[10] S. Erturk. Digital image stabilization with sub-image phase correlation based global motion estimation. *IEEE Transactions on Consumer Electronics*, 49(4):1320–1325, 2003. 3

[11] Wilko Guilluy, Laurent Oudre, and Azeddine Beghdadi. Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication*, 90:116015, 2021. 2

[12] Sang hun Jin and Gwang sik Koh. A robust image tracker based on phase correlation and fourier-mallin transform. In *2008 International Conference on Control, Automation and Systems*, pages 1028–1031, 2008. 2

[13] Jianwei Ke, Alex J Watras, Jae-Jun Kim, Hewei Liu, Hongrui Jiang, and Yu Hen Hu. Efficient real-time video stabilization with a novel least squares formulation. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1915–1919, 2021. 2

[14] Lakshya Kejriwal and Indu Singh. A hybrid filtering approach of digital video stabilization for uav using kalman and low pass filter. *Procedia Computer Science*, 93:359–366, 2016. Proceedings of the 6th International Conference on Advances in Computing and Communications. 2, 6, 7

[15] Sanjeev Kumar, Haleh Azartash, Mainak Biswas, and Truong Nguyen. Real-time affine global motion estimation using phase correlation and its application for digital image stabilization. *IEEE Transactions on Image Processing*, 20 (12):3406–3418, 2011. 2

[16] Ohyun Kwon, Jeongho Shin, and Joonki Paik. Video stabilization using kalman filter and phase correlation matching. In *Image Analysis and Recognition*, pages 141–148, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 2

[17] Xinke Li, Haofan Mo, Fan Wang, and Yongming Li. Real-time and robust video stabilization based on block-wised gradient features. *IEEE Transactions on Consumer Electronics*, 69(4):1141–1151, 2023. 2

[18] Shuaicheng Liu, Mingyu Li, Shuyuan Zhu, and Bing Zeng. Codingflow: Enable video coding for video stabilization. *IEEE Transactions on Image Processing*, 26(7):3291–3302, 2017. 3

[19] Yu-Lun Liu, Wei-Sheng Lai, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Hybrid neural fusion for full-frame video stabilization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2, 3

[20] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004. 2

[21] Yuzuki Mimura, Chang Qiong, and Tsutomu Maruyama. Acceleration of video stabilization using embedded gpu. In *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 52–59, 2022. 2, 6, 8

[22] T. Musarath, Unnikrishnan K. Supriya, and G. Sreelekha. Sub-block based global motion estimation for affine motion model. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 665–670, 2018. 2

[23] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021. 6, 7

[24] Said Ousguine, Fedwa Essanouni, Leila Essanouni, and Driss Aboutajdine. Motion estimation of aliased images using the phase correlation. In *Second International Conference on the Innovative Computing Technology (INTECH 2012)*, pages 170–173, 2012. 2

[25] Alejandro Reyes, Alfonso Alba, and Edgar R. Arce-Santana. Optical flow estimation using phase only-correlation. *Procedia Technology*, 7:103–110, 2013. 3rd Iberoamerican Conference on Electronics Engineering and Computer Science, CIIECC 2013. 2

[26] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV (1)*, pages 430–443. Springer, 2006. 2

[27] Ethan et'al Rublee. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 2

[28] Steven W. S. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, USA, 1997. 2

[29] Fatih Emre Simsek, Cevahir Cigla, and Koray Kayabol. Sompt22: A surveillance oriented multi-pedestrian tracking dataset, 2022. 6, 7

[30] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 28(5): 2283–2292, 2019. 2, 3

[31] Xiao Wang, Shaofei Zheng, Rui Yang, Aihua Zheng, Zhe Chen, Jin Tang, and Bin Luo. Pedestrian attribute recognition: A survey. *Pattern Recognition*, 121:108220, 2022. 1

[32] Jiyang Yu and Ravi Ramamoorthi. Learning video stabilization using optical flow. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8156–8164, 2020. 2, 3

[33] Zihan Zhou, Hailin Jin, and Yi Ma. Plane-based content preserving warps for video stabilization. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2299–2306, 2013. 2

[34] Yingying Zhu, Ye Liang, and Yanyan Zhu. The improved gaussian mixture model based on motion estimation. In *2011 Third International Conference on Multimedia Information Networking and Security*, pages 46–50, 2011. 1, 2