Visual Odometry based Omni-directional Hyperlapse

Prachi Rani[1], Arpit Jangid[2], Vinay P. Namboodiri[3], and K. S. Venkatesh[4]

[1] Department of Elecrical Engineering
Indian Institute of Technology Kanpur, Kanpur, India
prachi.rani03@gmail.com
[2] Department of Elecrical Engineering
Indian Institute of Technology Kanpur, Kanpur, India
arpitjangid999@gmail.com
[3] Computer Science and Engineering Department
Indian Institute of Technology Kanpur, Kanpur, India
vinaypn@iitk.ac.in
[4] Department of Elecrical Engineering
Indian Institute of Technology Kanpur, Kanpur, India
venkats@iitk.ac.in

Abstract. The prohibitive amounts of time required to review the large amounts of data captured by surveillance and other cameras has brought into question the very utility of large scale video logging. Yet, one recognizes that such logging and analysis are indispensable to security applications. The only way out of this paradox is to devise expedited browsing, by the creation of hyperlapse. We address the hyperlapse problem for the very challenging category of intensive egomotion which makes the hyperlapse highly jerky. We propose an economical approach for trajectory estimation based on Visual Odometry and implement cost functions to penalize pose and path deviations. Also, this is implemented on data taken by omni-directional camera, so that the viewer can opt to observe any direction while browsing. This requires many innovations, including handling the massive radial distortions and implementing scene stabilization that need to be operated upon the least distorted region of the omni view.

1 Introduction

With time, high quality video cameras becoming cheaper and the growth of social media platforms that give the opportunity to share videos have resulted in people making videos more often. Recently the popularity of 360 degree view cameras like Ricoh Theta has attracted people to shoot the videos of their fun activities like bike riding, running, mountain climbing etc. Cheap storage devices have made storage of long videos easy but it is quite time consuming to watch and navigate through such large videos. One simple way to watch these videos in a short time is to speed them up in order to create timelapse videos. This results in a watchable short video in case of stationary cameras. But with

camera motion, which is actually the case most of the time with hand held or head mounted cameras, this will heighten the apparent motion resulting in a nauseating jumble. Some adaptive fast forward approaches [14] try to adjust the speeds in different parts of the video so that stationary scenes are sampled sparsely while dynamic scenes are sampled densely. Some timelapse and hyperlapse approaches combined with video stabilization have also been suggested for conventional cameras.
Storyboard summarization based on keyframes selection is proposed in [17,18]. Video stabilization techniques can directly be combined with timelapse (before or after). In [4,12] the camera path is generated by estimating pose using feature mapping and this path is used to create a new smooth path for the camera for stabilization. In [10,11], based on structure-from-motion (SfM) they render the scene for a novel smooth camera path. Hyperlapse app from Instagram uses the approach of stabilization after uniform frame skipping [6] and for stabilization they use hardware stabilization approach of Karpenko *et al.* [7]. Hardware based stabilization approaches involve dampening of camera motion using mechanical means like inertial sensors (gyroscopes and accelerometers) with gimbals. As instagram's approach uses inertial sensor data, it cannot be applied to existing videos.
Non-uniform timelapse with varying skipping rate across the video as a function of scene content is proposed in [3]. The most sophisticated approach for hyperlapse creation is from Kopf *et al.* [8] which is based on SfM so computationally very expensive. Joshi *et al.* [5] came up with real time hyperlapse creation approach based on optimal selection of frames using frame matching and poses hyperlapse as a frame sampling problem by optimizing an energy function. Very recently a saliency based hyperlapse approach for 360° videos have been proposed [9]. They generate hyperlapse by optimizing over saliency and motion smoothness followed by saliency aware frame selection.
Our work is based on smart sampling of the input frames depending on a cost function which works upon the camera path obtained by visual odometry [16]. We include the following costs in the cost function: (a) pose change cost between frames, (b) smooth path deviation cost by fitting a smooth approximate path to the VO camera path, (c) velocity and acceleration costs. The idea is to sample more frames when pose change of camera is significant. This method can be used for normal cameras as well as for omni-cameras, but creating omni-summaries is different from normal in terms of handling massive radial distortion and applying stabilization on the basis of the least distorted spatial regions of input video.

2 Framework

The proposed algorithm can be divided in two main parts: Visual Odometry (VO) and selection of frames based on VO.

2.1 Visual Odometry for Omnidirectional Camera

Preparing images in 'Ready for VO' form: As VO is a feature matching based process, all the video frames must be undistorted. The Ricoh Theta S camera consists of two fish-eye lenses each giving a view of more than $\frac{4\Pi}{2}$ steradian. In recorded video, frames are depicted as two discs side by side, with each showing one hemispherical view captured by one lens. Since the view is captured as a spherical image, it is very much distorted at the edges of each of the discs, and is hence difficult to restore. Observing that the central part is least distorted, we crop the central part from either of the discs and undistort this part only. This cropped part is undistorted using D. Scaramuzza's OCamCalib toolbox for MATLAB [15]. We observed that central 400X400 part proved to be good enough in case of 1920X1080 resolution. In case of 1280X720 resolution this size reduces to 300X300. This is not a very strict restriction but optimal in the sense that the above stated size frames can be undistorted well and in the final cropped versoin enough number of robust feature points can be found for VO. If we crop a bigger portion, there will remain some distortion which can cause matching errors and if we crop smaller portion there may not be enough features in the final cropped frame for VO.
The manufacturing company of Ricoh Theta S provides an app RICOH THETA [2] which converts the image in the 2 disc form of into equi-rectangular form. This equi-rectangular form video can be used to look into any specific direction using the same app. This feature of the app makes available hyperlapse data in equirectangular form, which is very convenient.

Finding VO: Visual odometry (VO) is the process of estimating the egomotion of an agent observing the changes in images of the attached camera due to induced motion. Two consecutive camera frames at different camera positions are related by rigid body transformation $\mathbf{T}_{k,k-1} \in \mathbb{R}^{4\times4}$ given as:

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \mathbf{t}_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{R}_{k,k-1} \in \mathbb{R}^{3\times3}$ is the rotation matrix and $\mathbf{t}_{k,k-1} \in \mathbb{R}^{3\times1}$ is the translational vector. Our main task is to compute the relative transformations $\mathbf{T}_{k,k-1}$ and then to get the full trajectory of the camera by concatenating the transformations. Main components of the VO system are shown in Fig. 1(f). Our algorithm does not include last step of VO which is bundle adjustment. Rotation and translation are extracted from the essential matrix $\mathbf{E}$ by 2-D to 2-D motion estimation and $\mathbf{E}$ can be computed from 2-D to 2-D feature correspondences. Nister's five-point algorithm has been used for motion estimation [13]. After extracting $\mathbf{R}$ and $\mathbf{t}$, $\mathbf{T}_{k,k-1}$ is obtained and camera pose can be found incrementally as shown in Fig. 1(g):

$$\mathbf{C}_k = \mathbf{C}_{k-1}\mathbf{T}_{k,k-1} \tag{2}$$

2.2 VO based Hyperlapse Algorithm

We propose a cost function based on VO and choose frames which follow the target speed-up rate, yet keep their pose close to the smooth path and the pose change between consecutive frames is also minimal. The following costs are included:

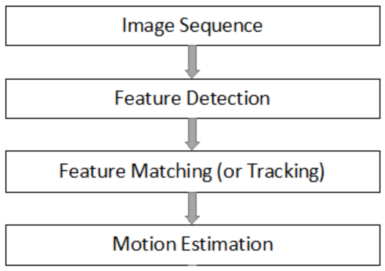
(a)

(b)

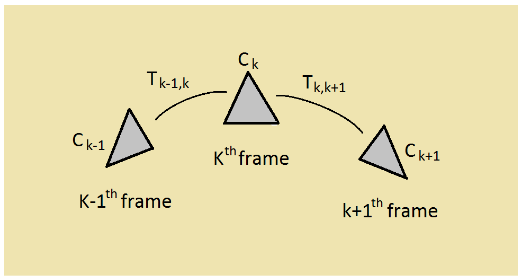
(c)

(d)

(e)



(f)



(g)

Fig. 1: (a), (b) are one omni-frames in two circle form and equi-rectangular frame. (c), (d) are 400x400 central cropped part and its undistorted version and (e) is 300x300 part used for VO. (f) is Block diagram of VO system used in our algorithm [16], (g) is illustration of VO process.

Pose Change Cost: One observation is that even minor camera pose change causes significant content change in consecutive frames (rotation causes more change than translation) assuming the environment is sufficiently diverse. So, in pose change cost, more weight is given to orientation change. Camera poses are obtained as 7D vectors (x, y, z and 4D quaternions) and 6D vectors (x, y, z and 3D Euler's angles (pitch, roll and yaw)). Pose change cost is computed as:

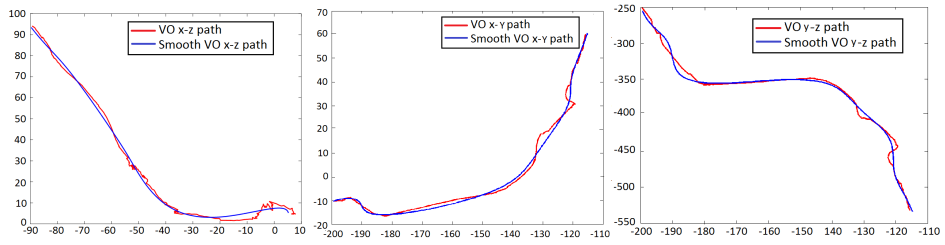
$$\begin{aligned} C_n(i,j) = &\|P_{xyz}(j) - P_{xyz}(i)\| \\ &+ Q_p * \|P_{pitch}(j) - P_{pitch}(i)\| + Q_y * \|P_{yaw}(j) - P_{yaw}(i)\| \\ &+ Q_r * \|P_{roll}(j) - P_{roll}(i)\| \end{aligned} \quad (3)$$

Here $P_{xyz}(j)$ is 3D translation coordinates, $P_{pitch}(j), P_{yaw}(j)$ and $P_{roll}(j)$ are the rotation coordinates of the j^{th} camera frame and Q_p, Q_r, Q_y are weights given to pitch, roll and yaw change respectively.

Smoothness Deviation Cost: For the resultant hyperlapse to look good in terms of frame-to-frame transitions, frames should be sampled such that they lie near to a smooth camera path. For the obtained camera path P as a sequence of 7D pose vectors, a smooth camera path S is obtained by fitting a cubic spline on the camera path obtained from VO as shown in Fig. 2 and the cost for j^{th} frame is defined as:

$$C_d(j) = \|P_{xyz}(j) - S_{xyz}(j)\| + w * \|P_{quat}(j) - S_{quat}(j)\| \quad (4)$$

Here $P_{xyz}(j)$ and $S_{xyz}(j)$ are 3D translation coordinates for actual and smooth camera path respectively, $P_{quat}(j)$ and $S_{quat}(j)$ are 4D quaternions vectors for actual and smooth camera path respectively for j^{th} camera frame.



(a) (b) (c)

Fig. 2: VO path and smooth path obtained by spline fitting.

Velocity and Acceleration Costs: As in [5], to ensure that our result achieves the desired speedup we use a velocity cost when the frame skip deviates from the target speedup v.

$$c_s(i,j,v) = min(\|(j-i)-v\|_2^2, \tau_s) \tag{5}$$

Here i, j are consecutive frames to be selected in hyperlapse, v is target speed up. Previous costs lead to selecting frames such that it balances between following a smooth camera path versus violating target speed-up. There can be cases where violating speed-up rate may lead to frames following a smooth path but it might cause a perceptual jump due to sudden change in speed-up rate. To avoid these sudden jumps in speed-up rate causing perceptual jumps, we use an acceleration cost:

$$c_a(h,i,j) = min(\|(j-i)-(i-h)\|_2^2, \tau_a) \tag{6}$$

Here h, i, j are consecutive frames to be selected in hyperlapse video, τ_s and τ_a are the upper bounds so that not a large number of frames are skipped.

Adaptive target speed up: Parts of the input video where there is sudden camera movement, at the same global speed-up result might be jerky at those places as we are not using frame-to-frame matching. Taking inspiration from equal motion hyperlapse of [5] we define adaptive speed-up depending upon the change in camera pose:

$$Q_d(i) = P_{quat}(i+1) - P_{quat}(i) \tag{7}$$

$$Q_{ds}(i) = f_{sigm}(\|Q_d(i)\|) \tag{8}$$

$$f_{sigm}(x) = \frac{1}{1+e^{-a(x-c)}} \tag{9}$$

$$v_{ad}(i) = \frac{median(Q_{ds})}{Q_{ds}(i)} \tag{10}$$

Here $P_{quat}(i+1)$, $P_{quat}(i)$ are quaternions vectors for actual camera path for $(i+1)^{th}$ and i^{th} frames respectively, a, c are sigmoid parameters and $v_{ad}(i)$ is target speed up for i^{th} frame.

Finding optimal path: Using all the costs, the total cost with different weights for different components is defined as:

$$\begin{aligned} C(h,i,j,v) = l_s C_s(i,j,v) + l_a C_a(h,i,j) + l_n C_n(i,j) \\ + l_d(C_d(i) + C_d(j)) \end{aligned} \tag{11}$$

For obtaining optimal path P, define the cost of a path p for a given target speed-up v as:

$$\Phi(p,v) = \sum_{t=1}^{T-1} C(p(t-1), p(t), p(t+1), v) \tag{12}$$

$$P = arg\ \underset{p}{min}\ \Phi(p, v) \tag{13}$$

Optimum path calculation is done using algorithm 1 which consists of two passes. The first pass builds a dynamic cost matrix $\boldsymbol{D_v}$ (function of target speed-up v). For i^{th} frame, the next frame is to be chosen from among the window of w successive frames.

Algorithm 1 : Frame selection

```
**Input:** $v$
**Initialization:**
**for** $i = 1$ to $g$ **do**
  **for** $j = i+1$ to $i + w$ **do**
    $\boldsymbol{D_v}(i, j) = l_d(C_d(i) + C_d(j)) + l_n C_n(i, j) + l_s C_s(i, j, v_{ad}(i))$
  **end for**
**end for**
**First Pass: populate** $\boldsymbol{D_v}$
**for** $i = g$ to $T$ **do**
  **for** $j = i+1$ to $i + w$ **do**
    $c = l_d(C_d(i) + C_d(j)) + l_n C_n(i, j) + l_s C_s(i, j, v_{ad}(i))$
    $\boldsymbol{D_v}(i, j) = c + min_{k=1}^{w}[\boldsymbol{D_v}(i - k, i) + l_a C_a(i - k, i, j)]$
    $\boldsymbol{T_v}(i, j) = c + arg\ min_{k=1}^{w}[\boldsymbol{D_v}(i - k, i) + l_a C_a(i - k, i, j)]$
  **end for**
**end for**
**Second Pass: track back min cost path**
$(s, d) = arg\ min_{i=T-g, j=i+1}^{T, i+w} \boldsymbol{D_v}(i, j)$
$\mathbf{p} = < d >$
**while** $s > g$ **do**
  $\mathbf{p}$ = *prepend* $(\mathbf{p}, s)$
  $b = \boldsymbol{T_v}(s, d)$
  $d = s$, $s = b$
**end while**
**Return:** $\mathbf{p}$
```

The algorithm populates $\boldsymbol{D_v}$ by iterating over its entries, where each element $\boldsymbol{D_v}(i, j)$ represents the running minimal cost path with last two frames as i and j. At each step of constructing $\boldsymbol{D_v}$, cost functions C_d, C_n, and C_s are evaluated at i and j, and the preceding frame h with the lowest cost is searched for, which depends on the previous costs and C_a. h is stored in a traceback matrix $\boldsymbol{T_v}$ for the second pass of the algorithm. After fully populating $\boldsymbol{D_v}$, the second pass finds an optimal path by finding the minimal cost in the rows and columns of $\boldsymbol{D_v}$.

3 Observations and Results

Since this is VO based process which is a feature matching based process, we can not use very low resolution cameras (either conventional or omni cameras like Ricoh Theta S) as low resolution frames increase the possibility of wrong feature matching which might ruin the accuracy of VO. For conventional cameras (GoPro Hero 4), we need not do any extra-preprocessing like cropping before undistortion as the inherent distortion is not too severe so the bigger full size frames can be used for VO. Using cropped frames for VO (relatively fewer features for matching) in case of the Ricoh camera does not change the VO quality much. We compare the odometry obtained from both cameras by matching different numbers of features after making a video capture with both cameras fixed rigidly to one another (so that their motions are identical, barring a small constant displacement), and choose that number for which the shape of VOs match the most. One more key observation here is that only the shape of VO matters, not their absolute values. On comparing the VO from both spheres, we observe that the shape is related irrespective of which disc we use for VO Fig. 3(b), thus confirming that the VO from any Ricoh sphere can undoubtedly be used for hyperlapse. The displacement in curves is observed because of the opposite direction of VO process accumulation error.
VO of raw video captured by head mounted camera can directly be used for hyperlapse, no unnecessary jerks are observed in hyperlapse but raw input videos captured by hand held camera are shaky due to induced vibrations in hand because of vehicle as in Fig. 3(a) so VO is not smooth. So VO of raw videos captured by hand held Ricoh Theta S can not be used directly for hyperlapse as unnecessary jerks can be observed in hyperlapse. In this case, a prior stage pf stabilization is applied before the VO as shown in Fig. 3(a).
We compare our method against Instagram's method (timelapse followed by stabilization) for conventional cameras by creating VO based hyperlapse with centrally cropped and undistorted frames. The measures used for comparison are the number of frames selected at sharp camera path points, mean and standard deviation of consecutively selected frames at these sharp camera path points. We also introduce a standard deviation based measure which is defined as follows:

$$measure = \frac{No.\ of\ non\ zero\ pixels\ in\ std.\ dev.\ image}{Total\ pixels\ in\ std.\ dev.\ image} \tag{14}$$

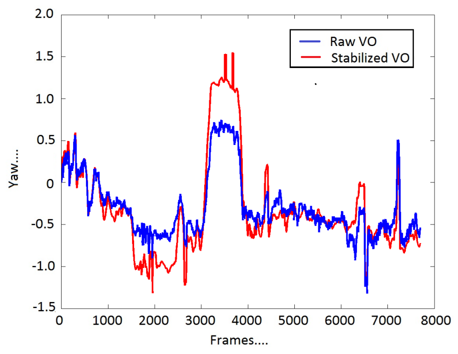
Better method:

1. selects more number of frames.
2. gives lesser ghostly mean and standard deviation.
3. gives lesser standard deviation based measure.

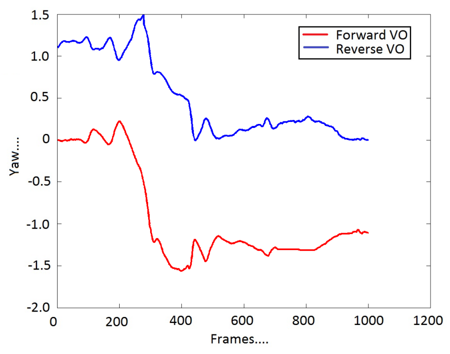
In table 1, we see that for our method, as compared to instagram's, more frames are selected at sharp turns and standard deviation based measure is also lesser. In Fig. 3, we see that mean and standard deviation for our method are less ghostly.

	Standard deviation based measure		No. of frames chosen	
	Timelapse+Stabilization	Our Method	Timelapse+Stabilization	Our Method
Slot1 at 3x	0.9870	0.9531	26	40
Slot1 at 6x	0.9934	0.9490	13	20
Slot2 at 3x	0.9527	0.9220	33	46
Slot2 at 6x	0.9749	0.9456	18	30
Slot3 at 3x	0.9997	0.9985	29	46
Slot3 at 6x	0.9986	0.9951	15	25

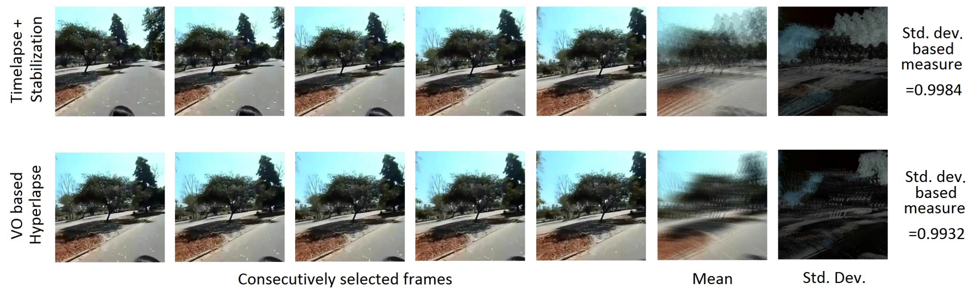
Table 1: Standard deviations based measure of 5 consecutive frames in case of 3x speed up and of 3 consecutive frames in case of 6x speed up is listed in table. This measure is defined as per equation 14. Also, number of chosen frames for different video slots at different speed ups have been shown in table.



(a)



(b)


Timelapse + Stabilization
Std. dev. based measure =0.9984
VO based Hyperlapse
Std. dev. based measure =0.9932
Consecutively selected frames
Mean
Std. Dev.

(c)

Fig. 3: (a): yaw curves for raw and pre-stabilized input video. (b): yaw curves from different spheres. (c): Consecutive selected frames and their mean and standard deviation. In case of VO based hyperlapse mean and standard deviation are less ghosted. We are comparing at a speedup of 3x with 'timelapse followed by stabilization' which is equivalent to instagram's hyperlapse technique. Numerical value of standard deviation based measure has been calculated as per equation 14.

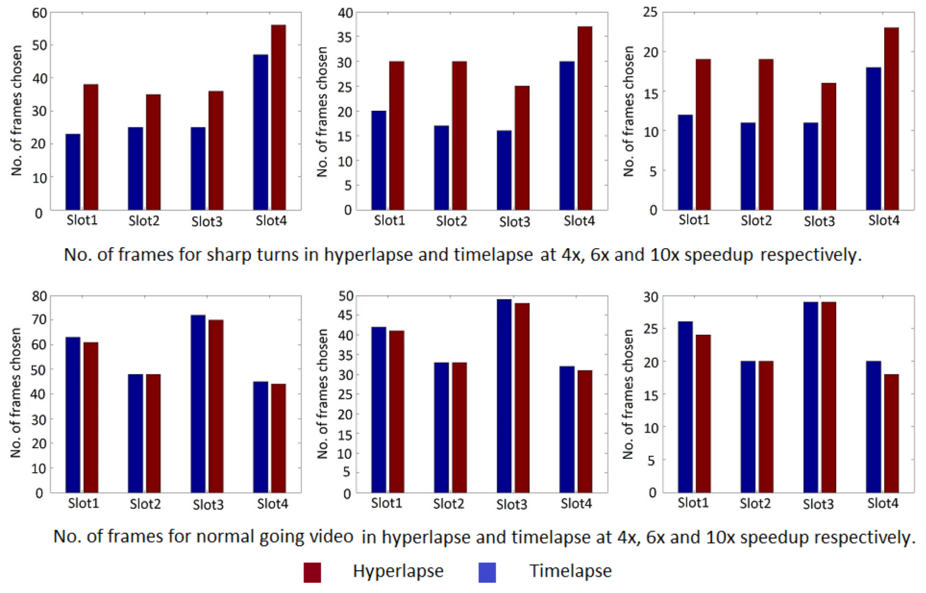


Fig. 4: Comparision of number of frames chosen in ours and Instagram's hyperlapse technique for different video slots at different speed ups. In case of normal going video number of frames selected in both are same which ensures that dense sampling takes place at sharp camera motion points only.

The poststabilization of omnidirectional hyperlapse is an issue in dual sphere format. Dual fisheye sphere format video is not accepted by RICOH THETA app for converting into equirectangular form (which is of main interest) after any modification. This app accepts only raw videos captured by Ricoh Theta S. So an alternative approach is to apply hyperlapse on equirectangular video which can be stabilized by facebook stabilization app for equirectangular videos [1] (to be launched in near future) and then we can use this stabilized hyperlapse to see into any specific direction using RICOH THETA app. Prestabilization for VO is done by combining all 'Ready for VO' frames into a video and stabilizing that sequence. This is equivalent to stabilizing all full frames, converting them to 'Ready for VO' form and then finding VO.

4 Conclusion

Our work of creating hyperlapse for egocentric videos is a novel approach based on Visual Odometry. Moreover, we implement our approach on omni-directional camera so that there is no question of losing information in any direction and user is also allowed to select post capture viewing direction. To handle the issue of egomotion we use cost function based approach and introduce various costs which helps the hyperlapse to become smooth. To handle omni-directional camera (excessive distortion) we introduce some pre-processing steps like cropping and undistortion. We also state the problems faced while trying to stabilize the omni-directional video.

5 Acknowledgment

This project is a part of postgraduate dissertation research work at Indian Institute of Technology Kanpur, India, financially supported by the Indian government organization *Defence Research and Development Organisation*.

References

1. 360 video stabilization: A new algorithm for smoother 360 video viewing, `https://code.facebook.com/posts/697469023742261/360-video-stabilization-a-new-algorithm-for-smoother-360-video-viewing/`
2. Ricoh theta app, `https://theta360.com/en/support/download/`
3. Bennett, E.P., McMillan, L.: Computational time-lapse video. ACM Trans. Graph. 26(3) (2007)
4. Grundmann, M., Kwatra, V., Essa, I.: Auto-directed video stabilization with robust l1 optimal camera paths. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. pp. 225–232. CVPR '11 (2011)
5. Joshi, N., Kienzle, W., Toelle, M., Uyttendaele, M., Cohen, M.F.: Real-time hyperlapse creation via optimal frame selection. ACM Trans. Graph. 34(4), 63:1–63:9 (2015)
6. Karpenko, A.: The technology behind hyperlapse from instagram, `http://instagram-engineering.tumblr.com/post/95922900787/hyperlapse`
7. Karpenko, A., Jacobs, D., Baek, J., Levoy, M., Virji, S.: Digital video stabilization and rolling shutter correction using gyroscopes (2011)
8. Kopf, J., Cohen, M.F., Szeliski, R.: First-person hyper-lapse videos. ACM Trans. Graph. 33(4), 78:1–78:10 (2014)
9. Lai, W., Huang, Y., Joshi, N., Buehler, C., Yang, M., Kang, S.B.: Semantic-driven generation of hyperlapse from 360° video. CoRR abs/1703.10798 (2017)
10. Liu, F., Gleicher, M., Jin, H., Agarwala, A.: Content-preserving warps for 3d video stabilization. ACM Trans. Graph. 28(3), 44:1–44:9 (2009)
11. Liu, F., Gleicher, M., Wang, J., Jin, H., Agarwala, A.: Subspace video stabilization. ACM Trans. Graph. 30(1), 4:1–4:10 (2011)
12. Matsushita, Y., Ofek, E., Ge, W., Tang, X., Shum, H.Y.: Full-frame video stabilization with motion inpainting. IEEE Trans. Pattern Anal. Mach. Intell. 28(7), 1150–1163 (2006)
13. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Trans. Pattern Anal. Mach. Intell. 26(6), 756–777 (2004)
14. Petrovic, N., Jojic, N., Huang, T.S.: Adaptive video fast forward. Multimedia Tools Appl. 26(3), 327–344 (2005)
15. Scaramuzza, D.: Ocamcalib: Omnidirectional camera calibration toolbox for matlab, `https://sites.google.com/site/scarabotix/ocamcalib-toolbox`
16. Scaramuzza, D., Fraundorfer, F.: Visual odometry [tutorial]. IEEE Robotics Automation Magazine 18(4), 80–92 (2011)

17. Xiong, B., Grauman, K.: Detecting snap points in egocentric video with a web photo prior. In: ECCV (2014)
18. Y. J. Lee, J.G., Grauman, K.: Discovering important people and objects for egocentric video summarization. CVPR 2(6), 1346–1353 (2012)