Removing Spinning Motion from Spherical Video Using a Rotationally Invariant Feature Extractor

Matt Slevin
School of Electrical and
Electronic Engineering
University College Dublin
Dublin, Ireland
matt.slevin@ucdconnect.ie
matt.slevin@arup.com

Paul Cuffe
School of Electrical and
Electronic Engineering
University College Dublin
Dublin, Ireland
paul.cuffe@ucd.ie

***Abstract*—This paper presents a method for removing rotational motion from an omnidirectional video captured from the viewpoint of a spinning object. For instance, the technique could be used to recover stable footage from spherical video captured from within a football used for competitive sports. The methodology uses a rotationally-invariant algorithm to obtain feature points and descriptors for pairs of successive frames. This information is then abstracted into three-dimensional point clouds, from which the Kabsch algorithm can infer the rotational motion between frames. The resulting rotational matrices are used to remap each equirectangular frame to a reference frame, and thereby offset the effect of frame-to-frame camera rotations. The algorithm has been successfully tested with various styles of footage. Additionally, the codebase implementing our algorithm has been freely shared through an open online repository.**

Keywords*—*Omnidirectional Video, Ball Camera, Video Derotation, Rotating Equirectangular Image

I. Introduction

The sports broadcasting industry is consistently striving to give its viewers the most satisfactory and immersive experience possible. In recent years, thanks to the emergence of robust action cameras, it has been possible for viewer to experience immersive action on the field. However the view from the point most central to the action has yet to be obtained: the view from inside the ball.

Traditional video cameras only enjoy a limited field of view. By contrast, omnidirectional cameras allow for a full field of view ($360^o \times 180^o$) to be captured. In recent years, consumer omnidirectional cameras have become more readily available and websites such as Youtube and Facebook offering free hosting of spherical video.

Neglecting the hardware engineering challenges of embedding a video capturing device into a ball, a primary issue with any footage so obtained is that the device itself would be spinning. The result is that the footage would be unwatchable. With omnidirectional video it is possible to simultaneously see in every direction and therefore reference objects remain visible in successive frames. The goal of the present project was to develop a technique by which omnidirectional video footage captured from the viewpoint of a spinning object (for example a football spinning through the air) can be decoupled from the rotational spin of said object, while preserving the translational motion. This paper outlines the approach used to achieve this goal.

Briefly, the developed technique derotates the video by firstly extracting image features between frames of the video. These features were matched, based on their descriptors, and used to calculate a rotational matrix that relates to the rotation of the camera between two pairs of frames. This rotational matrix was found by exploiting the Kabsch algorithm [1] [2]. These rotational matrices are then accumulated and used to determine the rotation trajectory of the object over time, with respect to the initial *anchor frame*. A remapping transform was applied to each frame to counteract the rotation of the raw equirectangular images. The output of this algorithm is omnidirectional video that maintains a fixed point of view even thought the capturing device exhibits rotational motion. The results of the present work can be recreated through the use of the publicly shared codebase [3].

A. Similar Works

Similar techniques exist for the stabilisation of omnidirectional video. One such technique utilises motion sensors in unison with scene tracking video [4]. Other solutions use purely software and computer vision techniques, [5], [6] being two such examples. These approaches aim only to stabilize video which is shaky, that is to say video which has minor motions along the three degrees of movement (pitch, roll and yaw). There is no evidence presented of these methods being tested with omnidirectional footage that has significant and constant spin, such as a ball in flight would experience. The testing of these methodologies in such circumstances is difficult without their code being publicly available.

However, there are three projects [7], [8], [9] which present their approaches to solve the present challenges of embedding a camera into a rotating sports ball in order to obtain usable footage. The work in [7] has presented a methodology which is quite similar to the approach used in this paper, as it likewise applies its algorithm to the video in post-production, with the same mapping projection being used. However, [7] uses an alternative matching algorithm and a different method to obtain rotation frame-to-frame, and is unclear on the precise method used to rotate each individual frames within the video. The present implementation emphasises open-source algorithms to allow code reusability and comparative analysis of results.

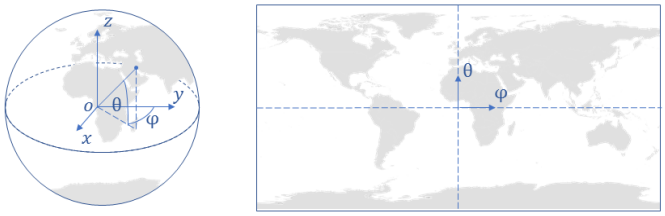

Fig. 1. Example of mapping a spherical representation of earth to an equirectangular projection.

B. Devices for Capturing Omnidirectional Video

There is a considerable variety of camera configurations capable of capturing omnidirectional videos, which vary widely in complexity, number of cameras, size and cost. The focus of the present research is on the processing of spherical video footage, rather than on how this can be captured. The video used as input to the algorithm has to meet certain criteria. Firstly, that the footage captured is fully omnidirectional (some camera configurations do not capture footage at the poles of the cameras POV). Secondly, each frame of the video should be stored as an equirectangular projection of the scene. This is the format that is currently used by the popular video hosting websites, such as Facebook and YouTube. Finally, any video used in the algorithm should have rotational motion. No sensor data is used and all data acquired was from the omnidirectional video using computer vision techniques.

II. Methodology

A. Equirectangular Projection

The map projection used to store the omnidirectional video is the *equirectangular projection*. This projection is widely used because it can be easily viewed through a viewer, which dynamically maps the projection onto a sphere with the viewer's POV located at the centre. This projection also fully depicts the surface of a spherical object, meaning it describes the full scene, $360^o \times 180^o$ degrees, captured by the omnidirectional camera. The type of projection can also be referred to as equidistant cylindrical projection, geographic projection, or la carte paralllogrammatique projection. The invention of the projection has been attributed to the Greek mathematician and geographer Marinus of Tyre some time in the period around 100 AD [10].

An example of this projection used on a spherical representation of earth can be seen in figure 1. The projection is used to map a 2D image onto a 3D sphere or vice versa. One useful property of the equirectangular projection is that the x and y coordinates on the map have a linear relationship with the longitude ϕ and latitude θ. This is particularly convenient from a computational perspective for mapping an image onto a sphere. The relationship between the spherical coordinates of the sphere and their Cartesian equivalents is described by equations 1, 2 and 3. Note these equations are defined for a 3D unit sphere.

$$x = cos(\theta)cos(\phi) \tag{1}$$

$$y = cos(\theta)sin(\phi) \tag{2}$$

$$z = sin(\theta) \tag{3}$$

Note that there are considerable distortions present in the equirectangular projection, particularly at the poles of the map and they are stretched to fit to the equirectangular map. This is an artefact of projecting a spherical surface onto a flat surface. This distortion at the poles is taken into consideration when performing feature extraction and matching as the raw projection does not accurately capture the scale of the scene in these regions.

B. Feature and Descriptor Extraction

Motion was tracked between successive frames using feature points and feature descriptors. Features are the points in an image that are visually distinctive. They can be identified between images via a keypoint descriptor, which presents a mathematical descriptions of the region about the feature points [11].

The feature and descriptor extractor algorithm selected for the present work was *Orientated Fast and Rotated Brief* (ORB) [12]. The algorithm was chosen for its vital properties of scale and rotational invariance. Scale invariance means that features and descriptor found in an image should not vary based on the scale of the image. Scale invariance is important due to the nature of the scaling distortions introduced at regions of high and low latitudes in the equirectangular projection. Additionally, rotational invariance is an essential property required from the ORB algorithm because the footage used in the algorithm will fundamentally have a significant rotational shift between each frame. The ORB algorithm uses the *Features from Accelerated Segment Test* [13] keypoint detector and the *Binary Robust Independent Elementary Features* [14] feature extractor.

It should be noted that ORB was chosen over SIFT [15] and SURF [16] for the purposes of this project because the latter two methods are patented algorithms, which hinders the availability and reproducibility of the algorithm. All three offer similar performance from a speed and accuracy perspective.

C. Feature Matching

Features were extracted for pairs of frames, a train and a query image, where the query image precedes the train image. The features between the two images were matched using a brute-force method. The descriptors of each keypoint were used for matching. Additionally, crosscheck was used to improve the quality of the results, by producing only a minimum number of outliers in the features matched. The matched features were then sorted in terms of descending order, based on the shortest distance between matched feature points of the image pair. The result was a list of sorted matches.

D. Data Abstraction

These sorted matches between the train and the query images were then converted into equivalent Cartesian coordinates. This was achieved in a number of steps. Firstly, the pixel location was converted into an equivalent longitude and latitude spherical coordinate values. Secondly, the spherical coordinates were then converted into corresponding Cartesian coordinates using equations 1, 2 and 3 to build a three-dimensional point cloud of image features. Only the best m

matches were stored (the choice of the value for m is discussed later) The result of this data abstraction was two point clouds, a point cloud for train image P_t and the query image P_q. Each row element of both point clouds P_t and P_q contains the location of the same feature point in three-dimensional Cartesian space for the respective frames. P_t and P_q matrices are identical in shape and size and have the following form:

$$\boldsymbol{P_q} = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_{m-1} & y_{m-1} & z_{m-1} \\ x_m & y_m & z_m \end{bmatrix} \tag{4}$$

E. Rotation Matrix

Each pair of point clouds can be used to infer a rotational matrix, which describes the cameras physical rotation between the train and query image. The method used to relate the point cloud pairs is the Kabsch algorithm [17], which calculates a (3×3) optimal rotational matrix U, which describes how to rotate one point cloud to best correspond to the other. The method can be summarised into the following steps:

1) Translation of both point clouds so that the centroid is located at the origin.
2) Computation of covariance matrix.
3) Computation of optimal rotational matrix U.

The algorithm requires that each of the inputted point clouds be of equal shape and size. An $(m \times 3)$ matrix was used. Each row of the respective matrices contains the coordinates (Cartesian coordinates) for a point. The number of rows, m, correspond to the number of feature points used and was a configurable parameter.

F. Accumulating Motion

Using the preceding techniques, a series of rotational matrices $\boldsymbol{U}$ has been obtained for each frame, as shown in figure 2. Each matrix describes the relative rotation between successive frames.

Using the same terminology as [9], the *anchor frame* is defined as the reference frame, and is used to relate all the other frames in the sequence via a rotational matrix. The anchor frame sets the viewing direction for all frames in the subsequent video sequence. For this project it was chosen as the first frame in the video sequence but any other frame in the sequence could be used. The rotational matrix that relates each frame of the video to the anchor frame was calculated by multiplying the preceding series of rotational matrices, as was calculated on a frame-by-frame basis. The resulting vector U_a contains rotational matrices. It is summarised in equation 5, where n is the number of frames in the video sequence. Each row element in the vector describes a rotational matrix U_{n-1} that can be used to relate the n^{th} frame to the anchor frame.

$$\boldsymbol{U_a} = \begin{bmatrix} U_0 \\ U_0 U_1 \\ \vdots \\ U_0 U_1 \dots U_{n-2} U_{n-1} \\ U_0 U_1 \dots U_{n-1} \end{bmatrix} \tag{5}$$

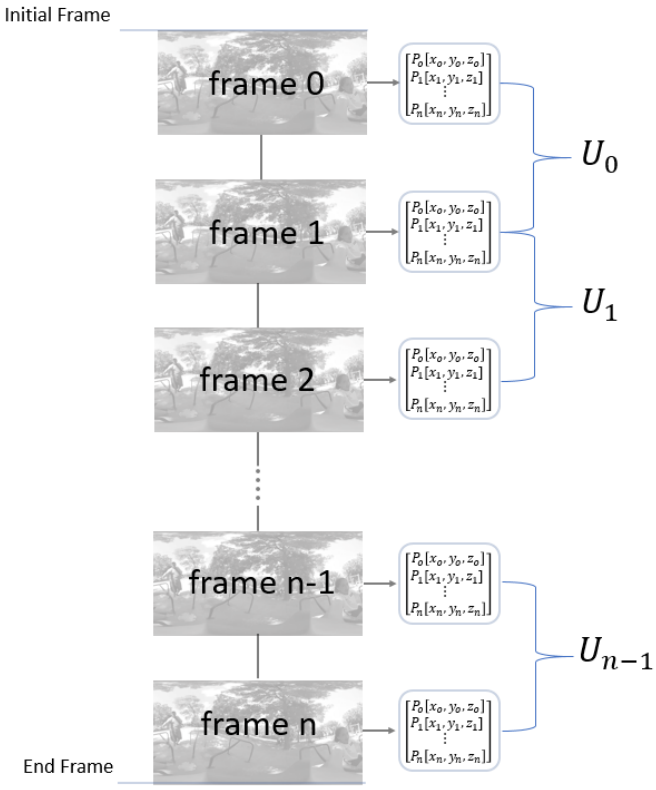

Fig. 2. Illustration of data acquired.

G. Transformation of Equirectangular Frames

Once the motion has been appropriately aggregated there exists a rotational matrix to relate each frame to the anchor frame. This rotational matrix can then be used to rotate the raw equirectangular frames to their corrected orientations. The steps used to rotate each frame within the sequence of images is as follows:

1) Each pixel in the equirectangular image was converted into its equivalent longitude and latitude value. The resulting values will vary significantly depending on the resolution of the image but had the same maximum and minimum range.
2) These longitude and latitude values are converted into Cartesian coordinates using equations 1, 2 and 3.
3) The three-dimensional coordinate representations of each pixel position are then rotated by multiplication with the relevant rotational matrix, to give a translated Cartesian coordinate for each pixel.
4) These values are then converted back their spherical representations using equations 6 and 7.

$$\phi = arctan(\frac{z}{\sqrt{x^2 + y^2 + z^2}}) \tag{6}$$

$$\theta = arctan(\frac{y}{x}) \tag{7}$$

5) Finally the RGB colour value of the initial pixel was saved into the new pixel location of the translated equirectangular position.

The use of interpolation was required on any of the resulting rotations that were not purely yaw (rotation about the vertical axes). The reason for this is due to the shift of an area with a low pixel density to an area with a high pixel density, for instance the centre region of the equirectangular projection where the pixel density is much higher than polar regions of the equirectangular image. This resulted in regions of missing pixels. Nearest neighbour interpolation is used to fill in these missing pixel values.

III. Results

The algorithm was implemented in Python (version 2.7.12) with the OpenCV libraries (Version 2.4.13). The main libraries used in this project were Numpy and Matplotlib for scientific computing and data visualisation purposes, respectively. Additionally, the Python Imaging Library (PIL) was used in assisting the remapping of the equirectangular projection.

A consumer omnidirectional camera was used to capture footage for testing purposes. The specific camera used was the LG 360 and was chosen for its relatively inexpensive price, full omnidirectional capturing capabilities and its small form factor. It captured two separate images using wide angle lenses for both sides of the device. These images were then stitched in order to produce an equirectangular image. The device captures footage with a resolution of 2560 $\times$ 1440 and a frame rate of 30 fps.

A. Number of Feature Points

For this implementation the parameter m which determined the number of points in the point cloud and features matched was set at $m = 40$. This was found to yield the best results. It was found that increasing m by an order of magnitude decreased the accuracy of the rotational matrix U obtained. The reason for this is that the number of outliers inputted to the Kabsch algorithm increased. It was also noticed that the run-time to obtain the rotational matrix also increased. When m was reduced by an order of magnitude the quality of the matched feature points returned increased, with fewer false positives. However, there was not enough points in the point clouds P_t and P_q for the Kabsch algorithm to return an accurate rotational matrix. More rigorous testing over a range of resolutions is required on the algorithm in order to determine the optimal choice of m.

B. Map Rotations

Examples of rotations applied to a single frame can be seen in figures 3 and 4. The sample image used for rotation is an equirectangular projection of a world map. Figure 4 depicts rotation about the y axis by $\frac{\pi}{2}$ radians. This process of rotating each equirectangular image is the most computationally intensive part of the present algorithm. It was noticed that as the pixel resolution of the inputted video increased, so too did the run-time of the algorithm. Even after vectorisation of the relevant code it is seen that approximately ninety percent of the total run-time is spent on rotating the frames of the video sequence.

Additionally, there is deterioration in the image quality after applying the rotation, with noticeable introduction of artefacts. A side-by-side comparison example is shown in figure 5. This illustrates the artefacts were introduced as a result of interpolation. This occurred due to the mapping of one region of the equirectangular projection which had a lower pixel density to a region with higher pixel density. Any rotations about the vertical axes provided the best results in the rotation of the image projection. The reason for this is because it only requires a horizontal shifting of pixels in the image. For this reason the best overall results attained from this algorithm are obtained when the input video only has rotations about the vertical axes. It would also be of benefit to use bilinear interpolation, as it would make the interpolation less noticeable.

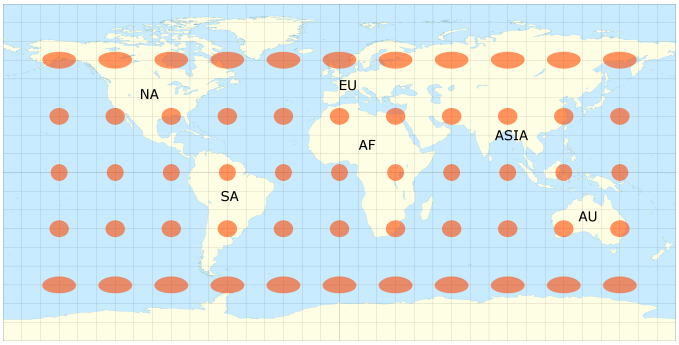

Fig. 3. Example of an equirectangular projection of earth, the orange ellipses illustrate how distortions increase at the poles of the projection (credit for this and derivative images: Eric Gaba / *Sting* under CC-BY-4.0)

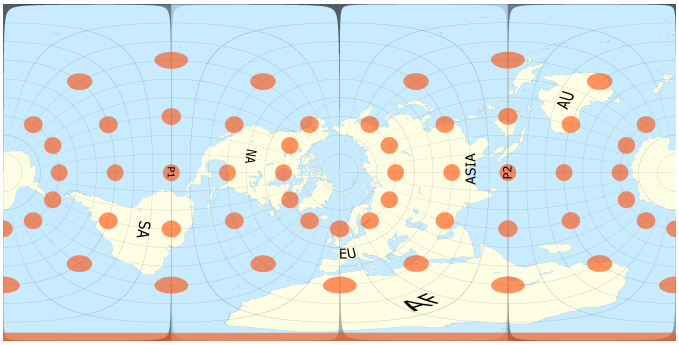

Fig. 4. Equirectangular projection rotated about the y axis by $\frac{\pi}{2}$ radians.

C. Test Footage

In order to validate the algorithm suitable test footage was required for different cases of usage. Various test videos were captured, and can be viewed in both their raw and corrected form at [18]. It is recommended to view these videos while reading this section of the paper. Two representative cases are presented here for analysis.

1) Test: Yaw With Translational Motion: A test was established to determine how the algorithm behaved in conditions where the camera experienced both yaw and translational motion simultaneously. The omnidirectional camera was hung from the roof of a tall agricultural building via a string. The camera was spun so that it rotated about the axis of the rope, and then pushed to mimic the swing of a pendulum. This can be visually seen in figure 6. The resultant footage captured both rotational and translational motion which aligned with the movement of the device.

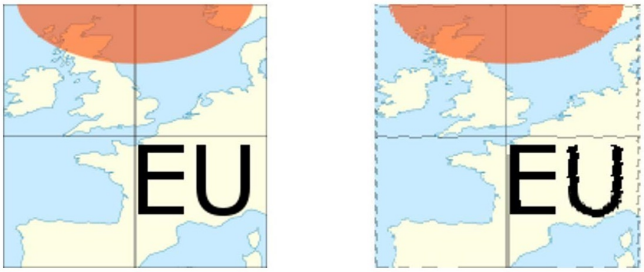

Fig. 5. Example of distortions introduced by nearest neighbour interpolation.

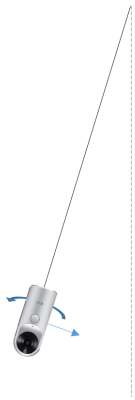

Fig. 6. Yaw with translational motion experiment

The resulting corrected video can be seen in figure 7. It corrects for the rotational motion by removing it but still preserves the translational component of the motion. The roof of the agricultural building can be seen as relatively stationary from frame to frame within the derotated video. The result of this test is paramount because it shows that the algorithm can work for omnidirectional video where there is both rotational and translation motion present in the recording device. Additionally, it shows that the algorithm works for footage where there is a change in the direction of translational motion.

An additional feature of the algorithm is the ability to infer the rotational velocity of the camera from U_a, the accumulated rotational motion vector. Arbitrarily defining the point $x = 0, y = 1$ and $z = 0$ as the initial starting orientation of the camera, it is multiplied by each of the rotation matrices in the U_a vector. This path is plotted and can be seen in figure 8. The plot demonstrates the rotational motion of the camera over time, indicating the start and end points. The video used to generate this graph was the same footage that was obtained by the experiment illustrated in figure 6. From the graph, it should be noted that the rate in change of colour overtime gives an indication to rate of change in speed. As the rate of change in colour remains relatively constant, it can be assumed that the speed of rotation is somewhat consistent as well. It can be remarked that the viewpoint of the user changes over time and that the motion is primarily about the z axis. This is an intuitive result based on the actual viewing of the footage.

2) Test: Thrown Camera: A test was run on the algorithm with footage that was captured from the camera while it was thrown in the air. The camera was thrown from one person to another with rotation of the device being random and about multiple axes. A visual illustration of the camera's motion is shown in figure 9.

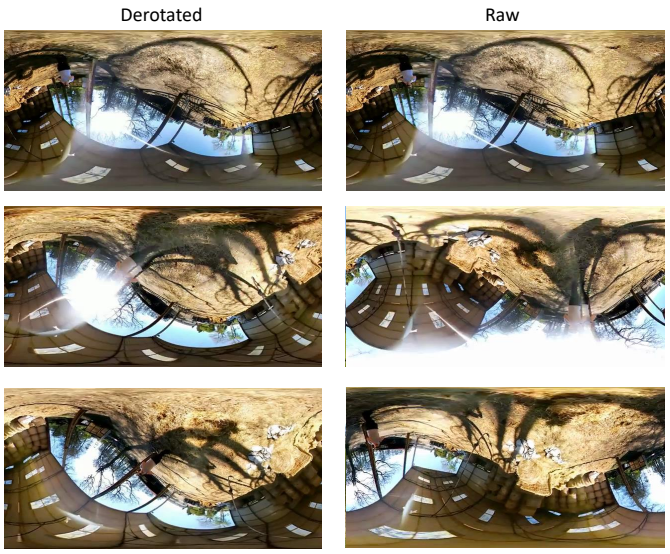

Fig. 7. Derotated footage vs raw footage from a camera with translational and rotational motion.

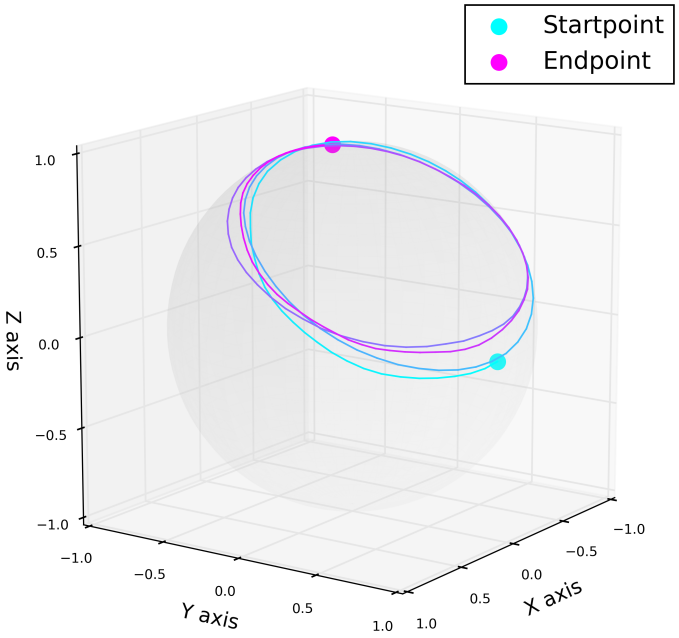

Fig. 8. Rotational trajectory of the camera for the footage used in the yaw with translational motion test

In this scenario, the algorithm failed to work satisfactorily after its initial run. However, the algorithm was able to correct some of the rotational motion within the video. The footage had significant roll pitch and yaw motion which could not be recovered with a single run of the algorithm. Each time the video was passed through the algorithm, the rotational motion of the camera was reduced. There was a significant noise component present in the form of rolling shutter effects. The camera was also moving rather fast through the air with motion not easily captured, making the reference objects move

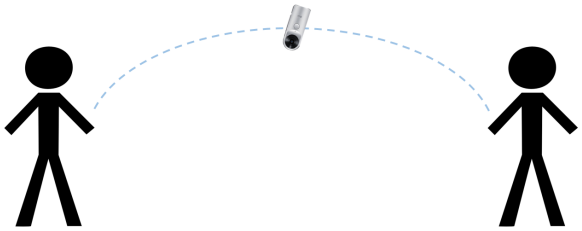

Fig. 9. Experiment illustration, the camera was passed by throwing resulting in random rotational motion

quite quickly relative to the camera. The result of correcting the video is illustrated in figure 10. Observing this it is clear that in the derotated footage there is a consistent viewing direction.

Furthermore, it is observed that the resulting corrected footage has a significant reduction in quality. The reason for this is due to the loss of information through interpolation, introduced with each iteration of the rotation algorithm. An improved implementation of the algorithm would be to obtain an accumulated rotational matrix over a number of iterations of the algorithm. This in turn would be applied directly to the raw video. This would not only result in a significant increase in output quality but also result in an improvement of the overall run-time of the algorithm.

IV. Conclusion

This paper outlined a method by which footage obtained from a rotating omnidirectional camera can be derotated. It used a rotationally invariant feature and descriptor extractor to match images between frames and obtain representative point clouds. From these point clouds the rotations between frames in the video can be inferred using the Kabsch algorithm. A method was outlined to use these rotational matrices to derotate each frame of the video. The underlying code for this algorithm can be viewed, edited and expanded upon through the public available repository. Although there is room for improvements in the algorithm the key methodology is outlined and it presents a novel solution to the problem of how to decouple footage from a camera embedded in a spinning object. It was also shown that this algorithm helps to reveal information in a scene that was previously unseen. It has potential applications in the entertainment industry to collect footage which is currently unobtainable, such as from cameras mounted within balls used for competitive sports.

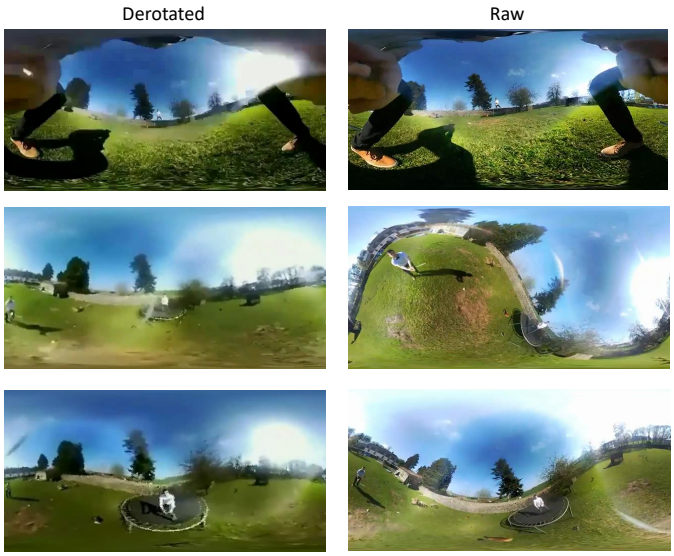

Fig. 10. Derotated footage compared with raw footage of camera being thrown through the air.

References

[1] Charnley, "charnley/rmsd," Dec 2016. [Online]. Available: https://github.com/charnley/rmsd

[2] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," in *Acta Crystallographica Section A*, vol. 34, 1978, pp. 827–828.

[3] M. Slevin and P. Cuffe, "360 video derotator," https://github.com/wkeu/360-video-derotator, 2019.

[4] T. Albrecht, T. Tan, G. A. West, and T. Ly, "Omnidirectional video stabilisation on a virtual camera using sensor fusion," in *Control Automation Robotics and Vision (ICARCV), 2010 11th International Conference on*. IEEE, 2010, pp. 2067–2072.

[5] J. Kopf, "360 video stabilization," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 195, 2016.

[6] S. Kasahara, S. Nagai, and J. Rekimoto, "First person omnidirectional video: System design and implications for immersive experience," in *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*. ACM, 2015, pp. 33–42.

[7] M. Nakazawa and H. Koike, "Synthesizing fixed point of views from a spinning omnidirectional ball camera," in *Proceedings of the 8th Augmented Human International Conference*. ACM, 2017, p. 32.

[8] R. Funakoshi, Y. Okudera, and H. Koike, "Synthesizing pseudo straight view from a spinning camera ball," in *Proceedings of the 7th Augmented Human International Conference 2016*. ACM, 2016, p. 30.

[9] K. Horita, H. Sasaki, H. Koike, and K. M. Kitani, "Experiencing the ball's pov for ballistic sports," in *Proceedings of the 4th Augmented Human International Conference*. ACM, 2013, pp. 128–133.

[10] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[11] M. Hassaballah, A. A. Abdelmgeid, and H. A. Alshazly, "Image features detection, description and matching," in *Image Feature Detectors and Descriptors*. Springer, 2016, pp. 11–45.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. IEEE, 2011, pp. 2564–2571.

[13] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[17] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[18] M. Slevin, "Thesis test footage," https://tinyurl.com/mkkaupg, 2018.