https://doi.org/10.2352/ISSN.2470-1173.2017.16.CVAS-342

Virtual tracking shots for sports analysis

Stuart Bennett[1], Joan Lasenby[1], and Tony Purnell[1,2]
[1]Cambridge University Engineering Department, Cambridge, UK
[2]British Cycling, UK

Abstract

Reviewing athletic performance is a critical part of modern sports training, but snapshots only showing part of a course or exercise can be misleading, while travelling cameras are expensive. In this paper we describe a system merging the output of many autonomous inexpensive camera nodes distributed around a course to reliably synthesize tracking shots of multiple athletes training concurrently. Issues such as uncontrolled lighting, athlete occlusions and overtaking/pack-motion are dealt with, as is compensating for the quirks of cheap image sensors. The resultant system is entirely automated, inexpensive, scalable and provides output in near real-time, allowing coaching staff to give immediate and relevant feedback on a performance. Requiring no alteration to existing training exercises has boosted the system's uptake by coaches, with over 100,000 videos recorded to date.

Introduction

Training for many sports is conducted over some fixed course, examples of such sports being athletics, rowing, cycling, skiing, and swimming. Being able to see, and record, what an athlete does throughout a training effort provides invaluable data for performance-analysts and coaches. In our particular case of cycling, aerodynamics are key, often giving the winning margin in competition, so a means of continuous capture permitting assessment of a cyclist's variable body position is of great interest. Figure 1 shows an example of wind-tunnel testing for posture enhancement.

Figure 1. *Aerodynamic 'envelope' highlighted on a static test*

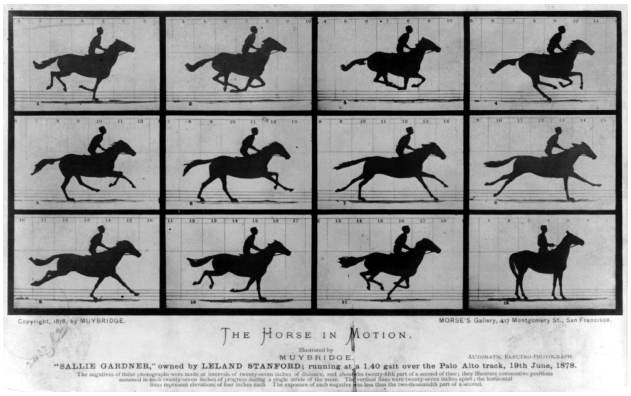

Figure 2. *The conceptual starting point: Muybridge's 1878 'The Horse in Motion'*

Capturing an individual's performance is challenging: static testing (as in Figure 1) imposes unrealistic constraints, affecting in-motion reproducibility; static cameras only provide potentially misleading snapshots of the whole effort; while rail-mounted cameras that physically follow an athlete are expensive to install and maintain, difficult to automate, and do not scale well to multiple concurrent athletes. Instead, we investigate reliably simulating a physically tracking camera through combining the output of many static cameras.

Multiple cameras have been merged over a course since the 1870s, with Muybridge's work on horse locomotion [1] (shown in Figure 2). We know of more recent work for sports, with a tracking shot for swimming demonstrated, but not productized in the years since. Our work's novelty lies in the entire system's automation, lack of expense, computational efficiency, scalability and immediacy. Riders' efforts are stitched on every lap, without any manual intervention or treatment, and the results are available to the coaches before the rider leaves the track, permitting tailored feedback to inform the next exercise.

Below we describe the design of our processing system, first dealing with the decentralized trigger, capture and analysis elements, before covering the centralized merging of footage to create a tracking shot. Following this we present example output and conclusions on the performance of our implementation.

System description

A critical issue in pursuing a many-camera architecture is cost. When hundreds of cameras are required to 'cover' a course, expensive machine-vision cameras are unlikely to be appropriate. Installing network capacity and multiple computers to receive and image-process each camera's high-bitrate stream could rapidly

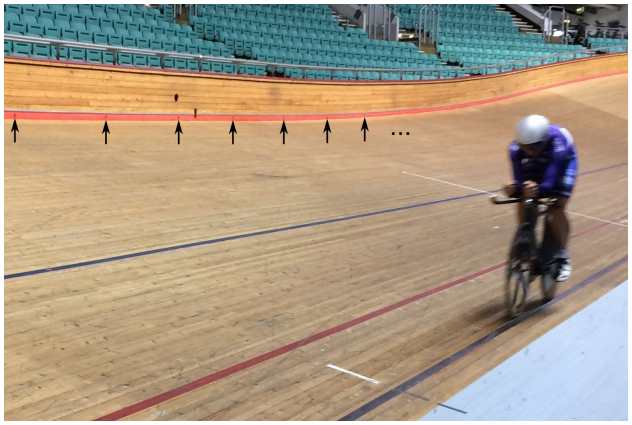

Figure 3. *Cameras (arrowed) installed at two metre spacing*

become expensive and scale poorly. Instead, a system where each camera is an inexpensive CMOS sensor attached to a commodity single board computer has been developed. Each node has a small but capable processor which allows it to autonomously perform the majority of the image-processing and analysis, without reference to its neighbours: detecting a cyclist passing, tracking the cyclist across the frame, and outputting a compressed video of the passing. Particular challenges include handling variable light levels, bicycle occlusion, and multiple riders travelling as a pack.

A single server can then take the per-camera clips and analysis results, which contain the timestamped positions of the detected bicycles, and from these positions infer the riders' tracks around the course. Relevant frames from each camera's clips are then combined and encoded into the stitched output tracking each cyclist separately.

The process described in detail below is summarized, and set in context, in Figure 7, a simplified flow-diagram concluding this section.

Camera modules

The camera-unit design was arrived at by considering a chain of constraints, detailed below.

Positioning

The desired view of the cyclist is that perpendicular to the bicycle/track, as in Figure 1. This is true both horizontally *and vertically*: a viewpoint looking down to the cyclist is less useful. Camera emplacements at the centre of the track can be liable to obstruction, and are vulnerable to damage, so the chosen location is low down on the track barrier.

Being therefore relatively near to the cyclists, wide-shots will fail the horizontal perpendicularity criterion: using images from the frame edges will have a front- or rear-facing component of the cyclist. The field of view used for output must then be relatively narrow, and the separation of cameras correspondingly reduced — empirically we found a two metre separation to give acceptable results, as highlighted in Figure 3.

Hardware

If the video is to be useful for analysis purposes, sufficient detail must be captured, which requires a resolution of around three pixels (px) per centimetre of cyclist: 600 – 700 pixels for a bicycle-length. For the camera module to trigger autonomously it's useful to be able to capture a frame or two before the bicycle is perpendicular to the sensor, adding at least another 300 pixels to both sides of the desired sensor width.

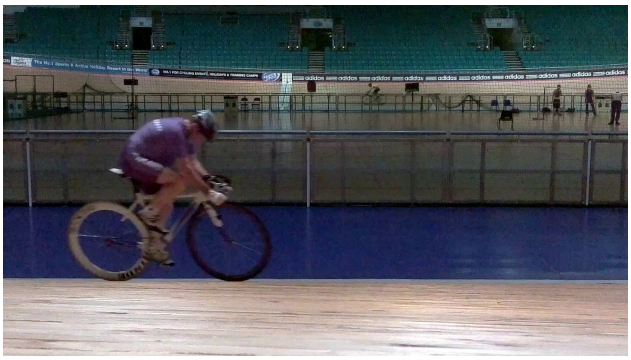

Figure 4. *A typical capture frame*

Clearly, considering common sensor sizes, a 720p (1280 × 720 px) sensor would *just about* meet the requirement, with a Full HD (1920 × 1080 px) sensor being far better suited. At present, larger sensors rapidly become much more expensive.

With cameras at two metre spacing around a 250 metre track, data management and hardware unit costs are major issues. If each camera were to stream to one or more servers, which in turn detect which frames have bicycles present, it's a lot of centralized computation, a lot of data being needlessly transmitted, and a lot of network provision to support that; all of which add to the cost.

The Raspberry Pi Foundation offers a Full HD camera module for use with their Single Board Computer ('RPi' SBC) [2]: a combination costing around £50 and coupling each camera with a capable multi-core ARM processor and GPU, both having access to the raw camera data. This means a camera node may then only upload cyclist-containing frames, using high-quality video compression, and need only be connected by cheap 100 Mbps network links to a gigabit network backbone. We use the camera variant without an infrared filter ('NoIR'), as the resultant increase in sensitivity makes up for the loss in colour fidelity, when the very short exposure times necessary to minimize motion blur are considered. The low power requirements of the nodes permit the use of Power over Ethernet, reducing installation costs and making each unit's power supply remotely manageable.

Image data

A typical frame from the viewpoints illustrated in Figure 3 is shown in Figure 4. The bottom of the frame is occupied by the track, the upper half has a variable background, capturing activities in the track-centre and on the other side of the track, while the cyclist passes across the middle of the frame, varying in scale depending on their lateral track position.

Capture trigger detection

To minimize processing and storage of unwanted data, we wish to only capture cyclists making a training effort, rather than an empty track, or those cyclists orbiting the track waiting for their turn or warming-down after an effort. To achieve this, video clips should only be saved when a bicycle is present around the red or black lines on the wooden boards.

The first step in determining this is for each camera node to self-calibrate where in its frame the wooden boards meet the inner blue-coloured area of the track, thereby allowing for variance in camera installation. Note that due to the shape of a velodrome track this edge may not be straight. By comparison of the sum of absolute differences between sequential frames (of the section of the frames expected to contain the board-edge), quiescent frames, without bicycles present, are found. Dividing the horizontal extent of the frame into forty blocks, for each block a threshold value is formed halfway between an average intensity taken from a patch in the blue-coloured area and a similar value from the wood-coloured area, thereby adapting to variation in lighting conditions (operations conducted on the red chroma channel ('V')), and the block is swept vertically to find the height of threshold transition. The forty edge estimates are then smoothed with an IIR low-pass filter, allowing the curve of the track-edge (if present) to be accommodated.

With the edge characterized, each frame is then 'straightened', with the image distorted such that the track-edge is entirely horizontal. A simple and very efficient action filter is applied to the left quarter of the current frame's luminance channel ('Y', in a YUV colour model), comparing the ten rows of pixels below the edge with both those in the previous frame and those in a recent frame in which no bicycles were detected (the 'background' frame). This will, intentionally, not respond to activity in the blue area above. Should either comparison fail to have a significant sum of absolute differences, a lack of triggering motion is presumed.

If triggering motion has not been discounted, we then perform further characterization of the motion. This involves comparing the current frame against the background frame. We look for one or more contact points made by black bicycle tyres with the track, and consider these contact points' height in the frame relative to the track-edge: any that are too low are too near the camera and so are discounted. A contact point must:

- be more than 25 pixels wide,
- at its edges be significantly different to the background in a band from the track-edge to three pixels below it, and
- across its width, with no discontinuities greater than five pixels wide, be significantly different to the background for a further ten pixels further down.

Furthermore, across the width of the contact point, and extending slightly from its edges, the three pixel band further beneath must not contain more than ten columns showing significant difference to the background, otherwise the contact is too near to the camera. This does not preclude other contact points being recognized, for instance during an overtaking manoeuvre.

Having found the contact point, the point itself taken as the centre of the extent described above, further filtering is performed against the blue chroma channel ('U'), looking just outside the expected locations of the leading and trailing edges of a nominal 700 mm diameter tyre against the blue area of the track. Comparing rectangles just before/behind the expected tyre against the background frame allows for ruling a contact point out if both rectangles are significantly different to the background, the implication being that the tyre is larger than expected and hence the bicycle is too close. Otherwise, a true contact point is found, and, if the point is in the left (bicycle entering) half of the frame, a recording is triggered.

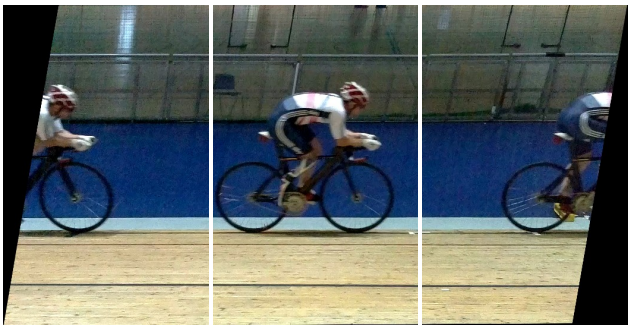

Figure 5. *Composite of three frames with rolling shutter corrected: perspective skew remains, away from the centre of the frame*

To keep the background frame current, when the below-the-track-edge regions of the current and previous frames have been similar for a number of consecutive frames, and the areas used in the 'U' channel test described above are similar for the current and previous frames (and the last background update was more than five seconds ago, or there has been a recent trigger), the stored background frame is updated.

Following a trigger frame, activity in the next six frames (allowing time for the cyclist to move entirely across the frame) is presumed and contact points searched for, without applying the action filter described above.

Contact point analysis and tracking

The 'U' channel wheel-size tests mentioned above serve another purpose. Of the rectangles before/behind the wheel, one should always show a significant difference — this being the bicycle frame — hence indicating a rear/front wheel respectively. These hints, along with contact point position data, are used to track wheels from one frame to the next, and deduce (pixel-based) bicycle velocities. These are useful data for the subsequent centralized clip stitching process.

Where two frames have previously been used to infer a velocity, the first simple tracking technique is to extrapolate expected wheel locations in a subsequent frame. Otherwise, noting that cyclists have upper-bounded positive velocities nearly always resolves associating wheels from frame to frame. Such approaches may obviously be extended by further extrapolation from more historic frames if a contact point was temporarily omitted (perhaps due to momentary occlusion) from a more recent frame.

Clip output

A downside of inexpensive CMOS camera modules is having 'rolling shutter', where the sensor scans out the image row-by-row, meaning the bottom row of a frame is captured at a slightly different time to that at the top. Hence in Figure 4 the wheels appear elliptical: as the bicycle moves during the capture the bicycle gets sheared. Armed with tracked velocities, this shear may be corrected, resulting in round wheels (as in the centre of Figure 5).

The wheels in the left and right parts of Figure 5 do not appear to be round however: due to perspective effects, circles will become ellipses oriented in one direction on the left of the frame, and in the other direction on the right. This perspective effect, if untreated, makes the stitched output visually discontinuous, as

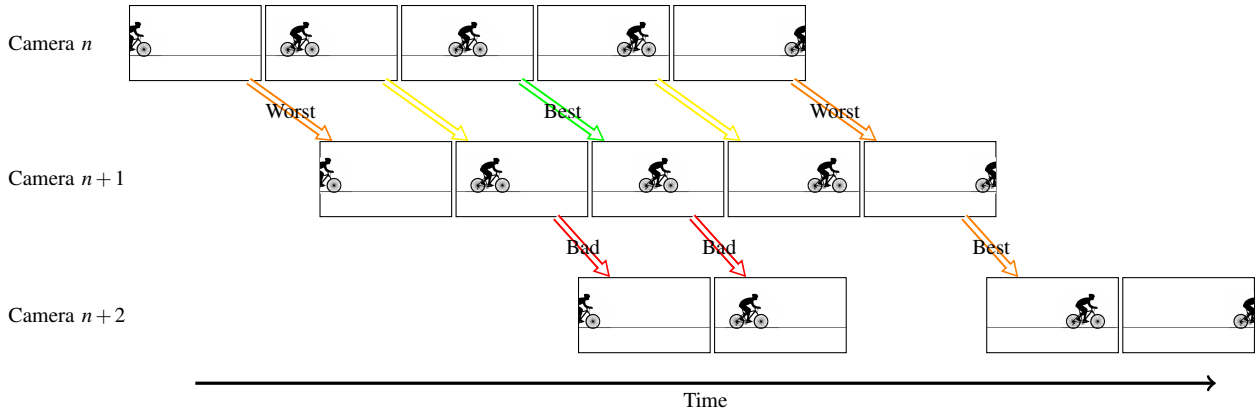

Figure 6. *Where available footage from a pair of cameras overlaps in time, we select frames to a) avoid skips and b) keep the bicycle close to the centre of the capturing frame. The transition between cameras n and $n+1$, while using poorly synchronized clips, has a substantial overlap and no missing frames, so only consideration b) applies. The second transition, while the clips are nearly synchronized, has to use a poor transition to enhance the chances of a transition to camera $n+3$. (Bicycles not centred in frames for clarity.)*

the output displays a bicycle taken from the right of one camera's frame and then from the left of the next camera's frame. The degree of skew is proportional to the position of the bicycle, so the tracking of the right-most wheel identified as a front tyre (the 'reference wheel') provides the necessary input to make the lead bicycle's wheels always appear round.

The position-based perspective-'correcting' component is added to the velocity-based shear coefficient, permitting the shear correction transformation to be performed on the RPi SBC, offloading a further task from the central server. As part of the shearing operation the clip is circularly shifted such that the reference wheel is located just right of the centre of the frame, so the bicycle as a whole is centred.

The relevant frames are H.264 encoded on the SBC, and the resulting timestamped clips and tracking metadata are then stored over the network, ready for stitching with the output of the other nodes.

Clip amalgamation

Due to the majority of the work being done on camera nodes, the central server only has three tasks:

1. associate tracks from one camera to the next,
2. select the frames giving a smooth progression of the cyclist along each track,
3. encode the frames to a make a merged output video.

Of these, the first is by far the most challenging. Naïve approaches, perhaps simply based around increasing trigger timestamps, are limiting. For example, with two cyclists travelling close together (as is common in 'pack' motion), one camera may make two separate clips, while another may merge them. Should the longer clip come from the second camera, there would be nothing to join the end of the second clip of the first camera; should the longer clip come from the first, the large difference in trigger times would make the clip an unlikely candidate to have immediately preceded the second clip of the second camera.

Always running a few seconds behind real-time (to allow clips and metadata chance to arrive from the camera nodes), we begin by loading the tracking metadata for each frame's reference wheel, and offset the analysed wheel locations by a pixel distance proportional to the capturing camera's index. This is necessarily imprecise, as the pixel distance presumes the cyclist to be at a certain depth from the camera, and so later matching must allow for the inter-camera gaps being over-/under-stated. Since a speeding bicycle has little acceleration or deceleration the progression of wheel locations against time are compared to a smooth fit, and any particularly poorly fitting points are substituted for interpolated values and the velocities adjusted accordingly.

Next comes the matching process: for each track in a given camera, can continuing tracks be found in the adjacent camera? This is impeded by both not having any unique identification of a bicycle, and the cameras' unsynchronized capturing. We first look for frames in the adjacent camera captured between half to one and a half frame-periods after the last frame of the track in the current camera, and, having found a candidate track, test all possible pairings between such broadly consecutive frames available between the two tracks. This exhaustive approach is possible as two cameras may both have a number of frames covering a given length of track, and useful as in marginal cases some pairings may not pass the pairing tests, while others do. The pairing tests consider whether:

- using the wheel location and velocity of the earlier frame, the predicted wheel location at the time of the later frame is close to the observed location, and
- the velocities recorded for the two frames are similar.

This rigorous consistency avoids accidental cyclist substitution part-way through a recording.

Should a pair of frames appear to follow on, the offset between the two tracks is therefore established, and the remaining question is one of frame selection for the output. Each possible transition is scored, penalizing solutions where frames omit the reference wheel heavily, and then favouring any where the bicycle is largely in the centre of the camera's field of view and hence (mostly) purely side-on (illustrated in Figure 6). The transition selected, the necessary frames are added to a 'chain', which is progressively extended as the matching process is repeated for each subsequent camera. Short chains, where some camera did not

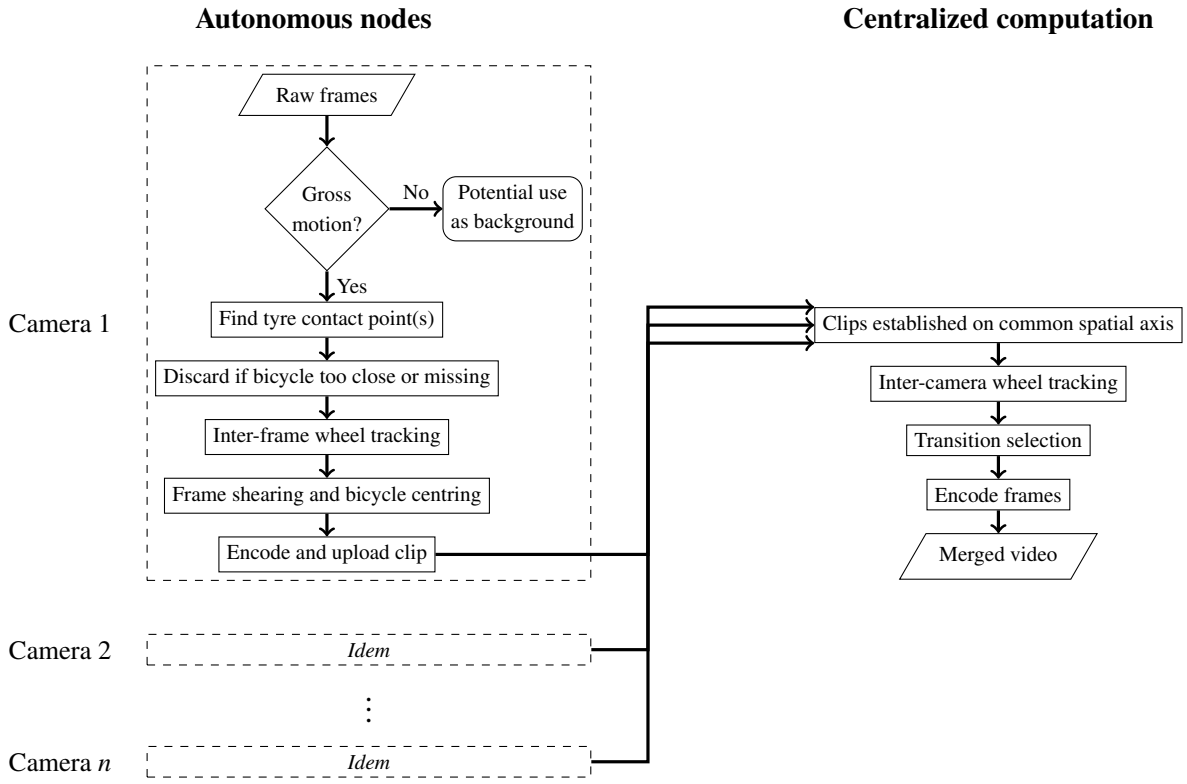

Figure 7. *A simplified flow-diagram summarizing the process pipeline*

capture a passing cyclist, are dropped. Note that this raises the bar for trigger reliability enormously: while it may seem satisfactory that a given camera may trigger 98 % reliably, requiring (say) ten to trigger in a row means that around one in five chains will get dropped, which is less satisfactory.

The chain formed, the source video clips are read to extract the frames specified in the chain; the frames are assembled and cropped to only show the centre, bike-containing, portion (assuming the transition selector has been able to source optimal frames this neatly crops the circularly shifted parts introduced during the shear transformation); and then encoded for output.

Example output

Sample output, merging four cameras, is shown in Figure 8. Counting the top-left image as frame one and proceeding rightwards then back to bottom-left, camera transitions occur between frames one and two, five and six, and eight and nine. In each case the change in apparent illumination/colour of the wood of the track may be observed, while the pedal angles advance smoothly. A less-well managed transition is apparent in the bottom-right frame, where the circular shifting is seen as an obvious discontinuity in the track-edge. Due to the rolling-shutter shear correction the wheels appear round, while the track-side railings seem skewed, as expected. The shape and position of the rider remains consistent and comparable, ideal for visual analysis purposes.

Our installation has created over 100,000 such output videos in entirely automatic operation.

Conclusion

We have described an affordable, capable and implemented architecture for delivering near-real-time virtual tracking shots. By using Raspberry Pi cameras and single board computers our per-node hardware cost is inexpensive (around £50). The hardware itself may be connected using commodity networking components and, critically, enables much of the computation to be offloaded from a central computing resource. Decentralized tasks of bicycle detection and tracking, and output clip deskewing, centring and compression are all undertaken on the camera node, leading to a substantially lower burden on the network and central server and much improved system scalability. The system has reliably generated over 100,000 videos in near real-time without requiring adaptation of venue lighting or training regimes, leading to good end-user buy-in by performance-analysts and coaches.

References

[1] O. D. Munn and A. E. Beach. A horse's motion scientifically determined. *Scientific American*, 39(16):241, October 1878.

[2] *Raspberry Pi 2 Model B and Pi NoIR Camera.* Raspberry Pi Foundation. URL: `https://www.raspberrypi.org/`.

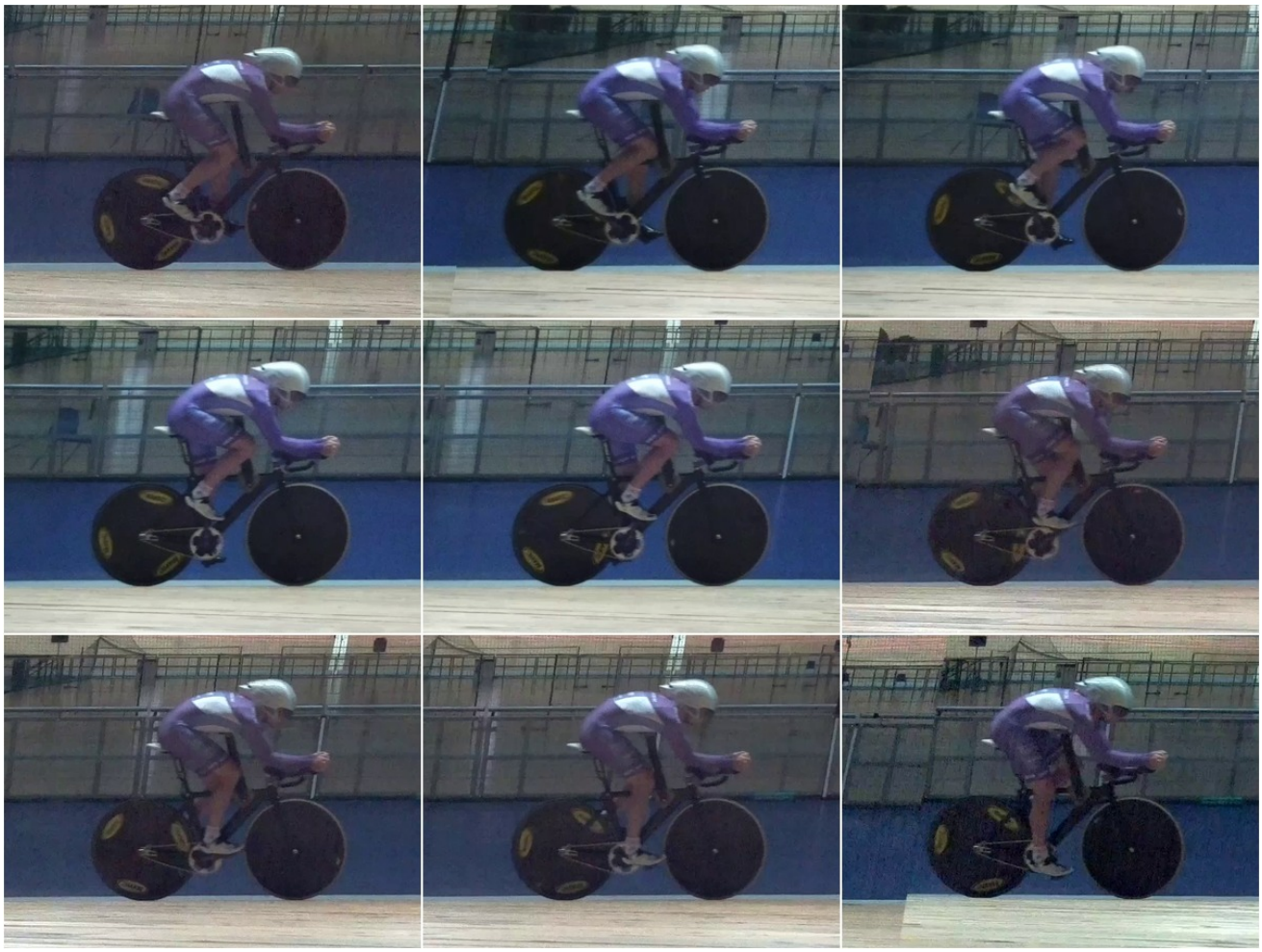

Figure 8. *Frame sequence from four cameras, processing left to right, top to bottom. Note changes in track colour between frames 1 & 2, 5 & 6 and 8 & 9 (different cameras) while pedals advance smoothly. Wheels are round, while background railings appear tilted, due to dynamic deskewing.*

Author Biography

Stuart Bennett gained his MEng in Information Engineering in 2007, and later his PhD on real-world 3D reconstruction in 2014, from the University of Cambridge, supervised by Joan Lasenby, in the Signal Processing Group of the Department of Engineering. Now a postdoctoral researcher, his research interests span computer vision, with a fondness for real-time robust image processing and multiple-view reconstruction applications with implementation challenges.

Joan Lasenby studied Mathematics at Cambridge University, then spent a year as a TA in Louisiana State University. Her subsequent PhD with the Radio Astronomy group in the Department of Physics, Cambridge, was followed by a Junior Research Fellowship at Trinity Hall, Cambridge, and working for Marconi Maritime Underwater Systems. Returning to academia, firstly as a postdoctoral researcher, then as a Royal Society University Research Fellow, she is currently a Reader in the Signal Processing Group of the Cambridge University Engineering Department. Her research involves image processing, motion capture, human motion modelling, medical applications of vision and geometric algebra.

Tony Purnell studied Engineering at Manchester University, before moving to MIT as a Kennedy Scholar, and then completing his PhD as a Benefactor's Scholar of St John's College, Cambridge. In 1986 he founded Pi Research, developing automotive analysis and control systems, sold to Ford in 1999. In the following years he has been principle of the Jaguar/Red Bull Formula One racing teams, and a technical advisor to the FIA. A Royal Academy of Engineering visiting professor at the University of Cambridge, in 2013 he became head of technical development at British Cycling.