Motion Based Remote Camera Control with Mobile Devices

Sabir Akhadov, Marcel Lancelle, Jean-Charles Bazin and Markus Gross
Department of Computer Science
ETH Zurich, Switzerland

ABSTRACT

With current digital cameras and smartphones, taking photos and videos has never been easier. However, it is still difficult to take a photo of a brief action at the right time. In addition, editing captured videos, such as modifying the playback speed of some parts of a video, remains a time consuming task.

In this work we investigate how the motion sensors embedded in mobile devices, such as smartphones, can facilitate camera control. In particular, we show two families of applications: automatic camera trigger control for jump photos and automatic playback speed control (video speed ramping) for action videos. Our approach uses joint devices: a remote camera takes a photo or a video of the scene and it is controlled by the motion sensor of a mobile device, either during or after recording. This allows casual users to achieve visually appealing effects with little effort, even for self portraits.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

Camera trigger; remote control; motion sensor; jump photo; video speed ramping.

INTRODUCTION

Taking photos and videos has become part of our daily life. Current cameras can capture high quality content in terms of pixel resolution, video frame rate and depth of field. Taking a photo or video is as simple as clicking a button. However, during acquisition a good timing can be difficult to achieve for a fast or brief motion, such as jump photos. In addition, editing a captured video, such as modifying its playback speed to emphasize an action (video speed ramping), can be time consuming and cumbersome. In this paper we investigate how motion sensors available on today's mobile devices (such as smartphones and smartwatches) can facilitate these two related tasks of camera control, i.e., automatic camera triggering and video playback speed control. We now discuss these two related tasks in more detail.

MobileHCI '16, September 06 - 09, 2016, Florence, Italy

ACM 978-1-4503-4408-1/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2935334.2935372

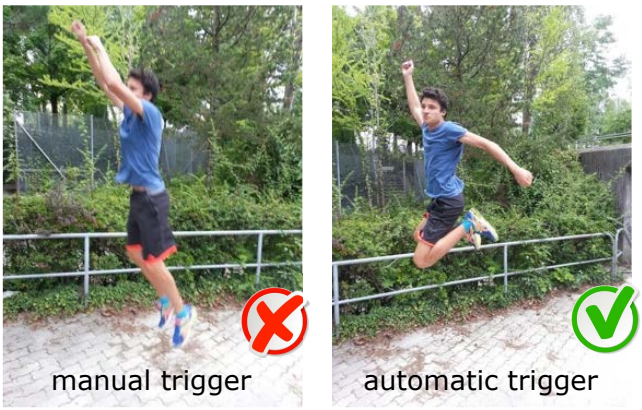


Figure 1: Manually triggering the camera for taking a jump photo is often error prone (see representative result on the left) and may require several attempts to obtain the desired photo in practice. In contrast, our approach *predicts* the time at which the highest point of the jump will be reached from motion sensor data and automatically triggers the camera at the right time, while taking camera delay into account (see representative result on the right).

Automatic camera trigger control.
Our work on camera triggering is motivated by jump photos. A jump photo is a great way to express fun and activity during a trip. However, taking a jump photo at the right time can be a challenge, and often multiple attempts are required (see Fig. 1). This is particularly true for smartphones due to relatively long trigger delays. We demonstrate an automatic trigger that *predicts* the time at which the highest point of the jump will be reached in order to take the photo at the correct time, while accounting for camera trigger delays. This is achieved with the motion sensors embedded in a mobile device carried by the jumping person. We leverage the motion data provided by the device to predict the time of the highest point and wirelessly send it to the remote camera (see Fig. 2). Our approach deals with any orientation of the camera, works in any lighting condition and even enables jump self portraits since it runs in a fully automatic manner.

Video playback speed control.
An additional family of applications with a related technical solution is an automatic playback speed control of videos (video speed ramping). This is motivated by the emergence of consumer grade slow motion cameras (such as the GoPro cameras and recent smartphones), especially for action videos.

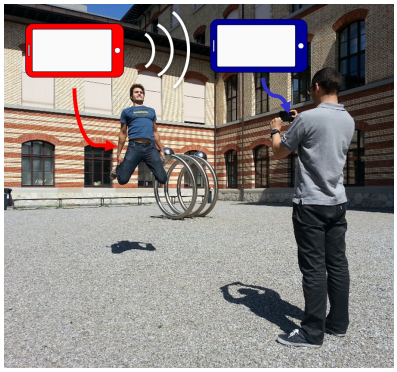


Figure 2: . Our approach leverages motion sensors embedded in a mobile device (such as a smartphone, in red) to control a remote camera (in blue), either live or after capture, depending on the use case.

Slow motion videos are entertaining to watch. Examples are popular videos with millions of views on YouTube[1]. However, the most interesting action might happen in just a fraction of the video duration, for example when a squash racket hits the ball. Obviously, it is not entertaining to watch a long slow motion video if nothing happens for most of the video duration. It means that the video playback speed should be controlled according to the action content. The video should be displayed at a normal speed when there is no action; and during the interesting action, the playback speed should be smoothly adjusted to achieve the desired slow motion effect. The usual way of changing the playback speed of parts of a video is to use video editing software, but this is time consuming and not an appropriate solution for casual users. In our work, the person performing the action (e.g., the squash player) wears a mobile device recording motion data, and we then use the motion information to automatically control the playback speed of the video acquired by a remote camera.

RELATED WORK

Automatic camera triggering can be achieved using additional non-commonly available hardware such as light barriers or pressure pads [8]. In contrast, our goal is to facilitate the process of taking jump photos without specialized hardware.

A solution could be to record a jump in video or burst mode and then *select* the frame when the jumping person is at the highest point. However two main issues exist. First, depending on the smartphone model, the video resolution and compression generally lead to an inferior quality compared to photos, and the frame rate in burst mode may be too low. The second main issue is the selection of the frame. An approach could be to let the user manually select the best frame but we aim for a fully automatic method. A related approach is the Adrenaline camera [2] that uses crowdsourcing to pick the best frames from a video by a majority vote. However this requires a crowd of online workers available 24/7 and this cannot provide results instantaneously because the image upload and the voting of the workers take time.

[1] https://www.youtube.com/user/theslowmoguys

An automatic approach for jump photos is the image analysis method of Maxwell-Parish [5]: it tracks the jumping person's face in a live stream video from a webcam on a computer, detects when the face location in the image starts falling back down and triggers the camera. However this approach has two main limitations. First, the photo will be taken too late due to the inevitable delays, in particular the camera trigger delay, which are especially noticeable in a smartphone context. Moreover, the face tracking and detection can easily fail in case of bad lighting, motion blur and occlusion. A similar image analysis method could be applied on a recorded video, rather than live stream, to select the frame of highest point, but this would also be sensitive to the errors of face tracking and detection. In addition, considering the time of the highest location of the face in the image (i.e., in pixel coordinates) as the time of the highest point of the jumping person is only valid for purely vertical jumps in front of the camera due to perspective effects. In contrast, our approach is applicable for any jump styles and any camera poses (see Fig. 6d).

To deal with the delays, an approach is to *predict* the time at which the highest point will be reached. Garcia et al. [3] track a person's face or a small patch on the view screen in real-time and calculate the person's velocity to predict the time of the highest point. They can thus trigger the camera at the right time while taking the camera delay into account. However, like the above method, this approach needs to track a person in the images, which is error prone in practice, and cannot deal with general jump styles and camera poses.

To overcome the issues of face tracking, jump style and camera pose, our approach leverages motion data from inertial sensors embedded in today's mobile devices. Such data has been shown to be effective to assist challenging computer vision tasks, such as 3D reconstruction [9] and video stabilization [1]. Our work explores how motion data can facilitate camera trigger control and playback speed control.

Various applications made use of motion sensors in the past with different goals in mind. As an example, the mobile application "Bump" [10] enables smartphone users to transfer data between two devices. The transfer is initiated when the users physically bump their smartphones together, and the bump is detected and identified with the motion sensor data. Pfeil et al. [6] demonstrate the triggering of a throwable panoramic camera ball using an accelerometer sensor. The user throws the camera ball in the air and the camera takes a panoramic photo when it reaches its highest position. However, the price of the current model is high for casual users and the motion sensor is attached to the camera. Instead, our goal is to take a photo of a jumping person, i.e., the camera should see the person and thus should not be attached to the person.

Video speed ramping (i.e., modification of the video playback speed) of action footage is a popular effect. However, only sophisticated video editing programs (such as Adobe Premiere) have this functionality and may require some training, which is not appropriate for casual users. We propose a second application of our motion sensor-based camera control to automatically adjust the playback speed of a video.

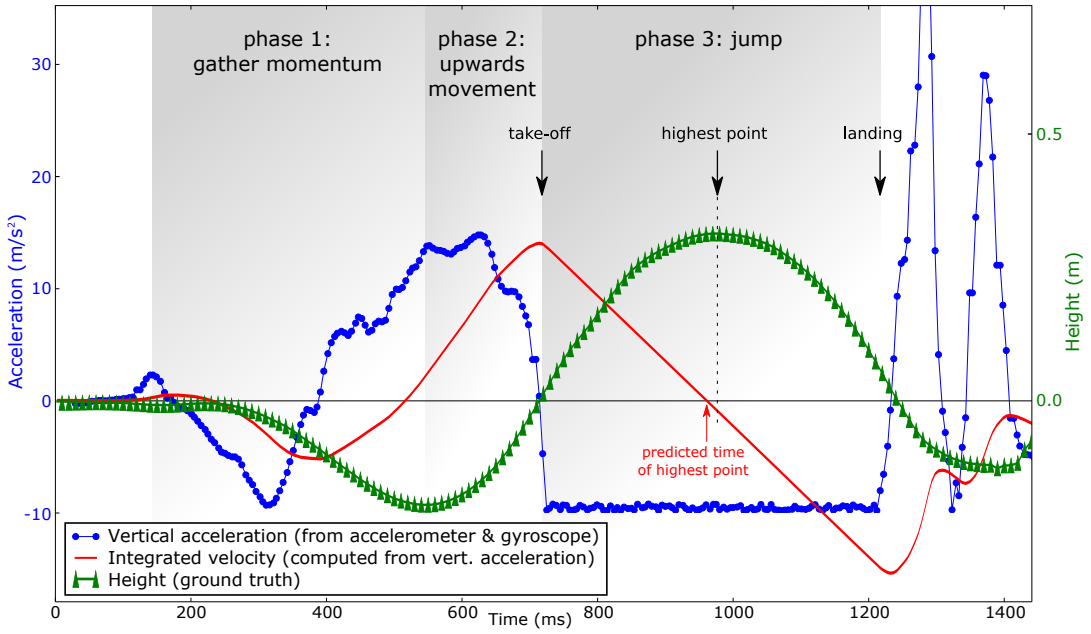


Figure 3: A jump may be divided into three phases. Phase 1: the person lowers to gather momentum. Phase 2: upwards movement until takeoff. At takeoff the vertical acceleration is integrated to compute the vertical velocity and the time of the highest point can be predicted (here with an error of about 15ms). Phase 3: during the jump the person is in a free fall state. The ground truth height is shown for illustration purpose and was obtained by manually annotating the frames of a video sequence which was simultaneously recorded during this jump.

PROPOSED METHOD

Automatic camera trigger

The key component of our automatic triggering method is to leverage motion data from the inertial sensors embedded in mobile devices to predict the time at which the highest point of a jump will be reached. Let's note v_0 the vertical velocity (i.e., along the gravity direction) of the jumping person at the moment of the takeoff. v_0 can be computed by integrating the acceleration of the device. Neglecting the air resistance, the motion of the jumping person during the jump phase is a ballistic trajectory. Therefore, the time of the highest point $t_{highest}$ can be computed by

$$t_{highest} = \frac{v_0}{g}, \tag{1}$$

where $g = 9.81 m/s^2$ is the gravity constant [11]. Fig. 3 shows the timing of a jump with the height, velocity and acceleration.

Computing the time of the highest point

In order to compute the velocity v_0, we measure the acceleration $\mathbf{a}_{measured}$ applied to the mobile device with the integrated three axis accelerometer. When the sensor is at rest, the direction of $\mathbf{a}_{measured}$ is towards the center of the Earth and its magnitude is $\|\mathbf{a}_{measured}\| = g = 9.81 m/s^2$. For our purposes we need the acceleration $\mathbf{a}_{relative}$ that causes the device to change its position. It is computed by

$$\mathbf{a}_{relative} = \mathbf{a}_{measured} - \mathbf{a}_{gravity}, \tag{2}$$

where the gravity vector $\mathbf{a}_{gravity}$ indicates the direction and magnitude of gravity. It can be obtained using sensor fusion from accelerometers and gyroscopes and applying a Kalman filter [4]. For our application running on Android, the developer API directly provides $\mathbf{a}_{gravity}$ (from the so-called *gravity sensor*).

For jumps executed mostly vertically, the major part of the measured acceleration is caused by the vertical motion. In the case of general jumps (e.g., forward jumps), the horizontal components can have a drastic impact on the predicted time and thus must be taken into account. Fig. 4 shows a significant difference between the vertical acceleration and the measured acceleration for a jump slightly forward. Therefore, to acquire photos of general jumps (i.e., not only vertical jumps), we need to know the vertical component of the acceleration $\mathbf{a}_{vertical}$. For this, we project the relative acceleration onto the gravity vector:

$$\mathbf{a}_{vertical} = -\frac{\mathbf{a}_{relative}^T \mathbf{a}_{gravity}}{\|\mathbf{a}_{gravity}\|_2^2} \mathbf{a}_{gravity}. \tag{3}$$

Then we obtain v_0 by integrating $\|\mathbf{a}_{vertical}\|$ before takeoff, during phases 1 and 2 (Fig. 4). Finally, we can compute $t_{highest}$ by Equation 1.

Transmission and camera delays

The predicted time of the highest point $t_{highest}$ is transmitted to the remote camera device. For that purpose, the mobile device with the internal motion sensor (i.e., carried by the jumping person) and the camera device are temporally synchronized before the jump. The Wi-Fi round-trip time of several messages is measured and the shortest one is used to store the

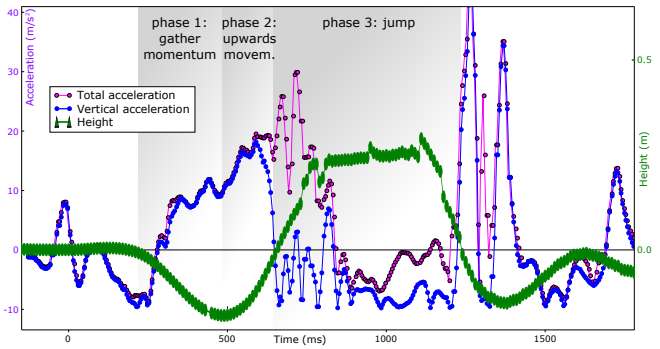


Figure 4: Acceleration of a mobile device in the pocket of a person jumping forward. The total acceleration (pink curve) is measured. The vertical component of the acceleration (blue curve) is derived by projecting the computed relative acceleration on the gravitational vector, showing a significant difference during takeoff.

offset between the internal clocks of the two devices. This approach allows to send the absolute time and thus avoid timing issues due to varying transmission delays. In our experiments, transmitting the value of $t_{highest}$, in absolute time, over Wi-Fi takes about 6ms.

The remote camera device is responsible to call the trigger function in advance, such that the exposure happens at the predicted time. The trigger delay can actually vary significantly due to focusing. That is why, in our implementation, the camera is focused right before the jump and auto-focusing is then turned off.

From the moment when the feet leave the ground to the peak of the jump, there is a duration of about 200-300ms, depending on the height of the jump. During this duration the prediction, transmission and camera triggering must occur.

Video playback speed control

Our method for video playback speed control builds upon our solution for camera triggering. A remote camera (in our case a GoPro camera) shoots a high frame rate video of an action and the person performing the action wears a mobile device recording motion data. The motion data and the video are synchronized by the timestamp. We recorded videos at 720p, 120fps. The video recording is started by the user from the mobile device. Once the video is taken, we use the recorded motion data to control its playback speed automatically. To prevent false detections such as from shocks or vibrations we apply a low pass filter on the magnitude of the input acceleration. In our experiments, simple thresholding of this filtered signal and local maximum extraction led to a successful detection of the interesting action moments (see Fig. 5). Each detected action moment is played in slow motion over a default duration of 2 seconds. The rest of the video is played at a normal speed. To obtain a smooth transition between the different frame rates, we use the smoothstep function. The playback speed change is instantaneous and different parameters such as motion sensitivity, slow motion time frame and the transition time between normal speed and slow motion

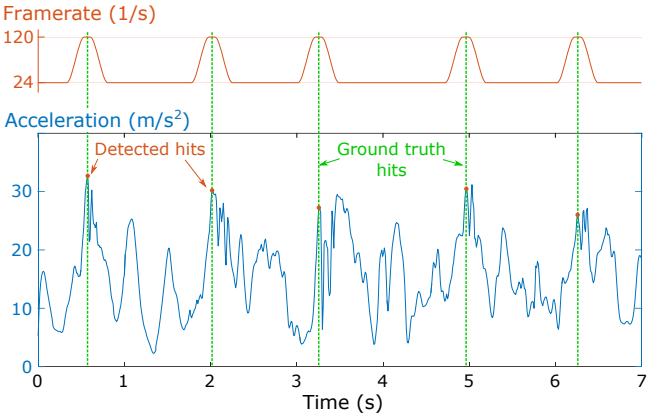


Figure 5: Automatic speed ramping of a video of a squash player hitting the ball. The measured acceleration (blue curve) of the squash player's arm is used to automatically detect the racket hits (orange dots) and control the video playback speed (orange curve) to show each hit in slow motion. The green dashed lines indicate the ground truth times of a hit, manually extracted from the video frames.

can be easily adjusted according to the users' preferences with intuitive sliders.

RESULTS

We now present the experimental results. We invite readers to visit our **project webpage**[2] for the supplementary video, additional results and our mobile app for smartphones.

Camera trigger for jump photos

We implemented our approach as a mobile app on two Android smartphones: one used as the remote camera and one as the motion sensor carried by the jumping person. Wearable devices such as smartwatches and activity trackers could also be used. The supplementary video shows how our app is used in practice: from device synchronization to the final jump photo. It shows that our app is simple to use, runs in a fully automatic manner and provides visually appealing jump photos.

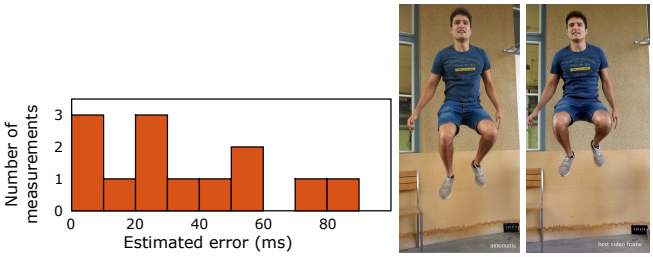
Fig. 6 shows representative results of jump photos automatically obtained by our app. One may note the range of camera orientations, lighting conditions, backgrounds and jump styles. All the results were obtained with a hand-held camera, except the self portraits of (c) with a camera on a tripod and (d) with a camera fixed on the backboard above the basketball hoop. Our method can also acquire photos of objects thrown in the air, see (g) where the mobile device is inside a plush toy. In the case of group photos (h), the camera is triggered when the jumping person carrying the mobile device reaches the highest point. We can also acquire entertaining photos. For example in (e), the persons jumped mimicking an explosion that was then composited with computer graphics content. Another example is the karate kick (b) where the person on the left stands still and the person on the right simply jumps vertically.

[2] http://cgl.ethz.ch/CamControl_MobileHCI2016

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6: Representative jump photos automatically acquired with our method.



(a) (b) (c)

Figure 7: Quantitative evaluation of our camera trigger method. (a) Distribution of the temporal error of our method with respect to the ground truth time. (b) Failure case where the photo is taken 85ms too early compared to the ground truth frame shown in (c).

To quantitatively evaluate our method, we used an additional high frame rate video camera for ground truth acquisition. From the high frame rate video, we manually selected the frame where the jumping person is at the highest point and considered it ground truth. The temporal error t_e is computed by $t_e = |t_{groundtruth} - t_{predicted}|$ where $t_{groundtruth}$ is the time of the manually selected ground truth video frame and $t_{predicted}$ is the time of the highest point predicted by our method. Out of 13 jump photos, 11 are timed with an error below 60ms (see Fig. 7a). The maximum error is less than 90ms and corresponds to the photo shown in Fig. 7b. The corresponding ground truth frame is shown in Fig. 7c.

Video playback speed control

The video results of our video playback speed control method are available in the supplementary material. They show that the desired effects can be obtained easily and successfully. For example, in the squash video, each hit of the ball is correctly detected and just before, during and after the hit, the video is played in slow motion, emphasizing the hit and omitting the less interesting time between the hits. While we only recorded casual scenes, we envision that our approach could also be used in professional sports training for an automatic video replay for feedback.

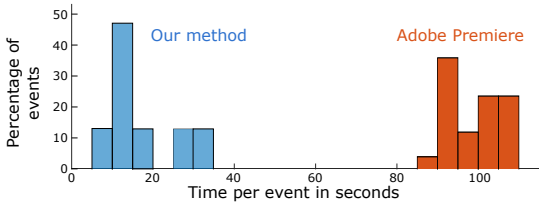


Figure 8: Distribution of the total user interaction time for speed ramping an event in a video, *including* the time for visual verification of the result.

To evaluate our method, we conducted a user study where we measured the duration needed to modify the playback speed of sport videos. The tested videos contain different types of activities (squash, parkour jumps, frisbee and ping pong), acceleration amplitudes and numbers of motion events per video. A total number of 63 events was used. Five users participated in the user study and were trained for both Adobe Premiere and our tool. For a fair comparison, we measured the total user input time for speed ramping *including* the verification time for visually checking the results of each motion event. The time for Adobe Premiere ranges between 85 and 110 seconds per motion event. It requires a lot of manual interaction and is complicated to use, especially for non-experts. In contrast, our approach is intuitive to use even for casual users and runs in a fully automatic manner. If needed, we allow the users to slightly adjust the motion detection threshold and the slow motion duration via intuitive sliders with instantaneous visual feedback. Our approach takes between 5 and 35 seconds per

event (including the verification time), which is much less than with Adobe Premiere (see Fig. 8).

Limitations and future work

A current limitation is the wireless connection range. We currently use Wi-Fi direct technology to connect the devices which has a working range of up to 100 meters. However, in our experiments a distance longer than 10 meters started to affect accuracy.

For video speed ramping, simple thresholding of the acceleration was sufficient to detect action moments in our experiments. A promising direction for future work is to use machine learning and pattern recognition to distinguish and select particular motions of interest [12, 7]. Acceleration data could also be combined with the rotational data from the embedded gyroscope sensor to detect and distinguish motions like acrobatic front and back flips.

The motion data from moving persons and objects can allow to perform other video related tasks such as video summarization, for example summarizing an hour long video down to the most interesting and action rich scenes only. We could imagine a scenario where a video surveillance camera, e.g., a car dashboard camera, uses a lower frame rate to save hours of passive recording and automatically saves a high frame rate sequence if the motion sensors detect some abnormal activity, for example a sudden deacceleration.

CONCLUSION

In this paper, we investigated how motion data can facilitate camera control. Our approach leverages the motion sensor embedded in today's ubiquitous mobile devices. We explored the two ends of the camera control spectrum: from the capture side by automatically triggering the camera, to the viewing side by adjusting the video playback speed.

On the capture side, our motivation was to acquire jumping photos at the right time, which is particularly challenging due to the brevity of the action. We demonstrated that we can use motion data from mobile devices to *predict* the time of the highest position and automatically trigger the remote camera at this predicted time with wireless communication. The experiments show that our method is accurate and can be successfully used in real world scenarios.

On the display side, we demonstrated that motion data can also be used to control the video playback speed. In particular, we showed how motion data can facilitate identifying the action moments of a video and automatically slow down the playback speed of these moments to emphasize them. The users can directly watch the enhanced video without having to use sophisticated video editing programs.

Finally, we believe that motion data from mobile devices opens many exciting research opportunities for visual content capture and processing, such as smart autofocusing, depth of field, camera motion control and visual effects.

REFERENCES

1. Steven Bell, Alejandro Troccoli, and Kari Pulli. A Non-Linear Filter for Gyroscope-Based Video Stabilization. In *European Conference on Computer Vision (ECCV)*, 2014.
2. Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2011.
3. Cecilia Garcia, Jean-Charles Bazin, Marcel Lancelle, and Markus Gross. Automatic Jumping Photos on Smartphones. In *IEEE International Conference on Image Processing (ICIP)*, 2014.
4. Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 1960.
5. Andrew Maxwell-Parish. Automatically Take Perfect Jump Shots. `http://www.instructables.com/id/Automatically-Take-Perfect-Jump-Shots` , 2013.
6. Jonas Pfeil, Kristian Hildebrand, Carsten Gremzow, Bernd Bickel, and Marc Alexa. Throwable Panoramic Ball Camera. In *SIGGRAPH Asia Emerging Technologies*, 2011.
7. Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity Recognition from Accelerometer Data. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2005.
8. Jason Poel Smith. Automatic Camera Shutter Switch. `http://www.instructables.com/id/Automatic-Camera-Shutter-Switch` , 2013.
9. Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco, Olivier Saurer, and Marc Pollefeys. Live Metric 3D Reconstruction on Mobile Phones. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.
10. Bump Technologies. Bump. `http://bu.mp/` , 2009.
11. Wikipedia. Projectile motion. `https://en.wikipedia.org/wiki/Projectile_motion` , 2004.
12. Yonglei Zheng, Weng-Keen Wong, Xinze Guan, and Stewart Trost. Physical Activity Recognition from Accelerometer Data Using a Multi-Scale Ensemble Method. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2013.