AUTOMATIC JUMPING PHOTOS ON SMARTPHONES

Cecilia Garcia, Jean-Charles Bazin, Marcel Lancelle, Markus Gross

ETH Zurich
Department of Computer Science
Universitätstrasse 6, 8092 Zurich, Switzerland

ABSTRACT

Jumping photos are very popular, particularly in the contexts of holidays, social events and entertainment. However, triggering the camera at the right time to take a visually appealing jumping photo is quite difficult in practice, especially for casual photographers or self-portraits. We propose a fully automatic method that solves this practical problem. By analyzing the ongoing jump motion online at a fast rate, our method predicts the time at which the jumping person will reach the highest point and takes trigger delays into account to compute when the camera has to be triggered. Since smartphones are more and more popular, we focus on these devices which leads to some challenges such as limited computational power and data transfer rates. We developed an Android app for smartphones and used it to conduct experiments confirming the validity of our approach.

Index Terms— Digital photography, automatic trigger, jump photos, mobile phones

1. INTRODUCTION

Jumping photos are very popular nowadays, especially for holidays and entertainement. However, a good timing is crucial to get a visually appealing jumping photo: if the camera is triggered a bit too early or a bit too late, the jumping picture is missed. Therefore, in practice many attempts are usually necessary to obtain a satisfying photo (see Figure 1). A self-portrait when jumping, with the help of a self timer or a remote control, is even more difficult. In this work, given the fact that smartphones are more and more ubiquituous, we focus on these devices, which leads to some technical challenges, as will be discussed in the following.

A simple solution to obtain a jumping photo with a smartphone is to record a video of the complete jump and later select the best frame from the video. However, compared to photos obtained by the camera mode of a smartphone, this yields a low quality result due to a lower resolution and stronger compression. Another solution is to use a camera equipped with a burst mode, capturing a sequence of photos, and later select the best result. However, the limited bandwidth of the camera results in a tradeoff between frame rate and image resolution, and moreover conventional smartphones do not have a burst mode. This means that with consumer-level smartphones, high-resolution photos can only be taken at a low frame rate, which drastically decreases the chance to capture the correct jumping picture.

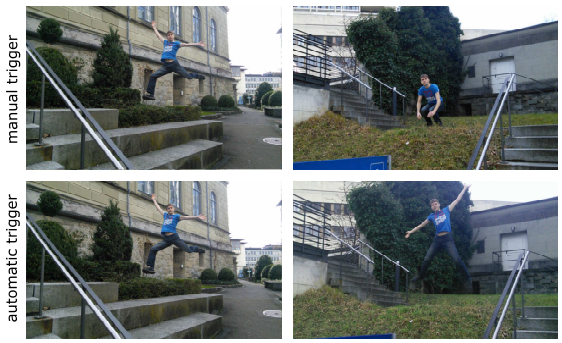

Fig. 1. The top row shows two jumping photos obtained with a camera manually triggered. A good timing is crucial: while manual triggering can sometimes yield good results (top left), it generally requires several attempts by trial-and-error (top right) The bottom row shows the same jumps automatically acquired with our Android app installed on a second device.

In order to facilitate taking a high-resolution jumping photo with consumer-level smartphones and with little effort, we propose a method for automatically triggering the camera at the right time to acquire visually appealing jumping photos.

2. RELATED WORK

Automatic triggering of a photo can be achieved with additional hardware. For a precise timing to trigger a photo of a moving object, light barriers are a common choice [1]. A camera can also be triggered with many other types of sensors, such as accelerometers embedded in the acquisition device to capture aerial views [2], [3]. However, all such approaches require additional hardware and too much setup time for our purpose of casual photos.

Automated triggering can also be performed via on-camera image processing. For instance, many of nowadays'

digital cameras have a smile detection feature to automatically take the picture when people smile. Visual motion detection has also been applied as a camera trap, for example, to take photos of lightning or wild animals.

To facilitate the development of computational photography apps, the FCam API was created for devices using the N900 hardware [4]. Unfortunately, this API does not support other hardware. Another option is the Frankencamera by Adams et al. [5], a custom camera platform allowing full low-level control of this camera. In contrast, our goal is to create a program that can run on existing conventional smartphones.

Recently and independently from our work, Maxwell presented a face detection-based approach to acquire jumping photos [6]. However, the result is a low-resolution video frame from a webcam. Moreover, the system triggers the acquisition of the final result when the tracked face starts falling back down and thus might miss the highest point. In contrast, our method (1) *predicts* the time of the highest point to achieve good timing and (2) returns a high-resolution photo.

Over the last few years, several programs based on computer vision techniques have been developed for smartphones, such as 3D reconstruction [7] and panoramic stitching (e.g. panorama tool in Microsoft's Photosynth). However, to the best of our knowledge, no app exists for automatically triggering high-resolution photos of jumping persons.

3. PROPOSED APPROACH

The main idea of our approach is to compute the optimal triggering time in advance. To achieve this, we analyze the ongoing jump online at a fast rate and predict the time at which the jumping person will reach the highest point. Taking the trigger delay into account, we compute the time when the camera must be triggered. All these steps are performed in real-time and in a fully automatic way.

To estimate the time corresponding to the highest point, we track the jumping person and fit a motion model to the trajectory when he/she is in the air. Hence, the algorithm needs to know when the ballistic phase starts, i.e. when the feet leave the ground, and use only the following measurements to fit the ballistic trajectory [8]. Detecting this transition is not trivial. To solve this problem, our method is based on the observation that the trajectory of a jumping person has a particular pattern that our algorithm uses to detect four distinct phases (Figure 2). In the following, we will first discuss how to obtain this trajectory and how to detect the phase transitions. Finally, we explain how to fit a motion model to the trajectory during the ballistic phase and how to estimate the optimal triggering time.

3.1. Jump trajectory

To obtain the motion trajectory of the jumping person, we detect and track the face, which is generally suitable for the targeted application of jumping person photos. Nevertheless, during the jump, motion blur is usually noticeable and thus the face detection might fail. Therefore, after the initial detection of the face in the first frame, we extract the most "salient" feature within the area of the detected face [9] and track it in the next frames. In terms of implementation, for the face detection, we use the approach proposed by Viola and Jones [10], for the feature extraction, we apply the feature detector by Shi and Tomasi [9] and the tracking is performed via KLT [11]. We also apply the autofocus on the face position at the first frame and then keep the focus fixed as autofocusing may be too slow and unreliable during the jump. Despite the simplicity of the approach, experiments showed that our tracking is robust and also accurate. A representative trajectory is shown in Figure 2. We assume that the distance between the jumping person and the camera does not change significantly during the jump. We only use the vertical (y) positions of the tracked face since the horizontal (x) values are not needed to predict the time of highest point.

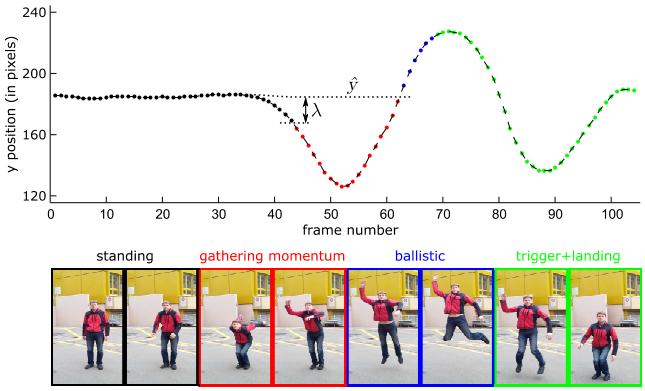

Fig. 2. Jump trajectory and jump phases automatically detected by our method (from left to right: standing, gathering momentum, ballistic and trigger+landing).

3.2. Jump phases

Our algorithm segments, in real-time, the jump trajectory into four phases (Figure 2): *standing*, *gathering momentum*, *ballistic* (person is in the air) and *trigger+landing*. Note that this last phase is for illustration purpose only: the final photo is already triggered during the *ballistic* phase and no more measurements are acquired after the camera is triggered. As soon as a face is detected, we enter the *standing* phase and assume that the person stands relatively still. At each new processed frame, we compute $\hat{y}$, the median of the y positions obtained from the start of the *standing* phase. If $\hat{y} - y < \lambda$, with y being the vertical position at the current frame and λ a threshold in pixels, then we switch to the gathering momentum phase. In all the results shown in this paper, we set λ to 6% of the image height, i.e. about 16 pixels in low-resolution 480×270 images. Experiments have shown that our method is not sensitive to the value of λ as long as it is meaningful, i.e. between 2% and 10%. In the next frames, once $y > \hat{y}$, we consider the

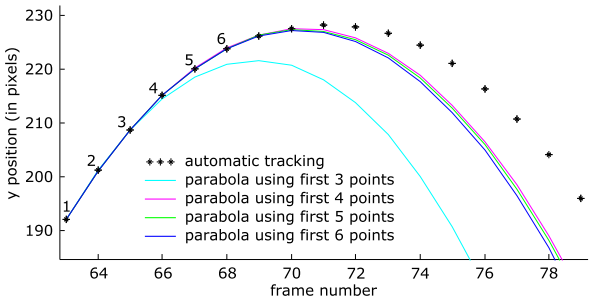

Fig. 3. Ballistic trajectory fitting with an increasing number of measurements from automatic tracking for the jump sequence of Figure 2.

feet do not touch the ground anymore and transition to the *ballistic* phase. A representative result is shown in Figure 2.

3.3. Ballistic curve fitting

To predict the time of the highest point, we fit a ballistic curve to the measured points of the *ballistic* phase. As the air resistance can be neglected, the curve is approximated by a parabola [8]. The parabola is represented by $y = \alpha_1 t^2 + \alpha_2 t + \alpha_3$ where y is the vertical position, t is time (or the frame index) and $(\alpha_1, \alpha_2, \alpha_3)$ are the parabola coefficients. A representative example with an increasing number of measurements considered for the fitting is shown in Figure 3. The time at which the highest point is reached is $t' = -\alpha_2/(2\alpha_1)$.

3.4. Optimal triggering time

A minimum theoretical number of three data points is needed to fit the parabola. Thus, as soon as three measurements are available, t' could be predicted. In practice, more measurements permit to have a more robust and accurate estimation of t'. Therefore, we repeat the process (parabola fitting and computation of t') with the newly obtained measurements as long as there is enough time to wait for a next measurement according to the current prediction. This is discussed in details in the following.

Once t' is computed, some steps must be performed before taking the final jumping photo. First, the camera mode needs to be switched from low-resolution (LR) to high-resolution (HR) capture mode. There is also a delay between calling the capture function (similarly to pressing a physical trigger button) and the start of the image exposure. Note that the computation is performed with a latency due to image readout. We refer to the sum of the latency and delays as the trigger delay Δ_d. Thus to take the picture at the computed time t', the camera must be triggered at $t^* = t' - \Delta_d$. The trigger delay Δ_d is hardware dependent and can be determined in advance, either by simple measurements/timers or from the technical specifications of the phone camera. If

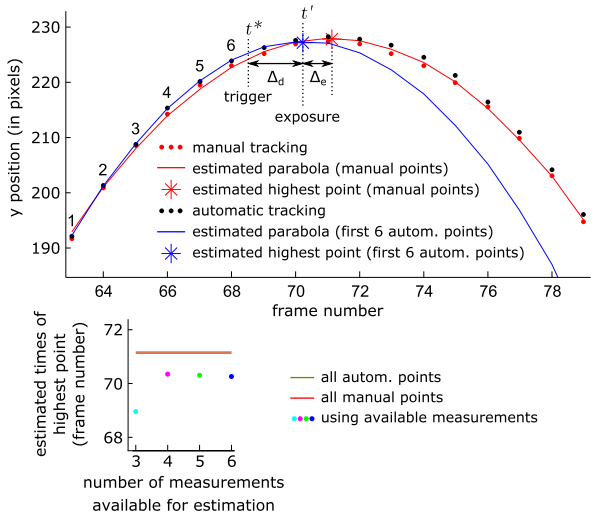

Fig. 4. Triggering time estimation for the jump sequence of Figures 2 and 3. Top: comparison of the estimated and ground truth time of the highest point (blue and red crosses), and illustration of the trigger time and delays. Bottom: the dynamically computed estimated time for the highest point.

needed, the estimated delay value can be adjusted by the user for his/her own smartphone.

Let Δ_i be the time interval between two images (acquisition and processing). At the current acquisition time t, if there is not enough time to wait for the next image/measurement, i.e. if $t + \Delta_i > t' - \Delta_d$, the acquisition of the final picture is prepared (e.g. switching from LR to HR mode). After waiting until t^*, the image is triggered so that the exposure happens at the computed time t', as illustrated in Figure 4. For the depicted sequence, six measurements are used to fit the final parabola and thus to estimate t' and t^*, since waiting for a next seventh measurement will not permit to trigger the camera on time.

4. RESULTS

Our approach is implemented as an Android app and deployed on multiple smartphones. The user launches our app, frames the shot via the displayed preview and then simply needs to click on a "start" button. Then, the processing is performed on low-resolution images in real-time and in an automatic manner, and captures a high-resolution picture of the jump. In case the face detection does not work, e.g. for a person not directly facing the camera, the user can tap on the smartphone screen to manually select the face or another textured part of the jumping person that is then tracked automatically. As an example, on our Sony Xperia smartphone, the jump is analyzed in low-resolution 480×270 at an acquisition and processing rate of around 25fps. Switching the camera from LR to HR mode to take the final jumping photo takes about 9ms and the estimation of the total trigger

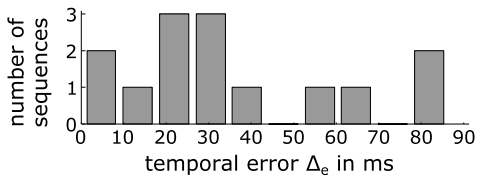

Fig. 5. Distribution of the temporal error of our method with respect to the manually estimated best time.

delay Δ_d is 49ms. The final jumping picture is acquired at a high resolution of 5 Megapixels (2592×1944).

The number of available measurements for the computation of the triggering time depends on the acquisition and processing duration Δ_i, the trigger delay Δ_d and the jump duration. From four to eight measurements were available in our experiments. The observed trajectory curve is not a perfect parabola because of some noise sources like hand-held camera motion, tracking inaccuracy and jumps slightly towards or away from the camera leading to perspective scaling of the y coordinate. To measure the influence of these factors and the temporal accuracy of our method, we conducted a quantitative comparison with ground truth data, as shown in Figures 4 and 5. The ground truth time of the highest point is obtained by manually tracking the face in each image and selecting the measurements of the *ballistic* phase (see red dots in Figure 4). Then we fit a parabola (red curve) to all these measurements and compute the time of the highest point as described in Section 3.3 (see red star in Figure 4). The temporal error Δ_e is computed as the difference between the ground truth time and the time estimated by our method. Figure 5 shows the distribution of the temporal error on 14 representative sequences. More than 70% of the tested sequences have a temporal error of less than 40ms. It shows that, in practice, despite the potential noise sources mentioned above, our method can still estimate the triggering time accurately.

Figure 6 shows several additional representative results for different scenarios and applications. All results are obtained by a hand-held device, unless otherwise stated. Note the different jump styles and amplitudes, as well as the different lighting conditions. Figure 6(e) shows a self-portrait obtained with a camera set on a stable location (here a wall). Beyond faces and persons, our app can handle various kinds of objects: the user simply needs to tap on the object to track on the smartphone screen. For example, Figure 6(f) is obtained by selecting the hippo plush toy that is then tracked automatically. The plush toy is thrown in the air and the resulting picture is taken when it is at its highest point. Figure 6(g) shows that our method can create visually impressive picture in a simple way. In this example the person on the right stands in a side kick position and the person on the left simply jumps vertically while mimicking being kicked. This jumping person is tracked by the app and the picture is automatically taken at his highest position. Figure 6(h) shows an example of a group of persons jumping together. In our current version, a single person is tracked: the final picture is taken when this person is at the highest position.

Fig. 6. Representative results automatically obtained by our smartphone app.

Despite the numerous visually appealing jumping photos we acquired, currently, the proposed method still has some limitations. The main one is that the face tracking can fail, especially in low light conditions due to severe motion blur. In standard daylight conditions, the face tracker worked generally very well. In addition, the KLT tracking can fail when the tracked point gets occluded during the jump, for example when an arm passes in front of the face.

5. CONCLUSION

In this paper, we proposed a method to automatically take high-resolution jumping photos with smartphones. The key idea of the proposed method is (1) to predict the time at which the jumping person will be at the highest point by analyzing his/her trajectory in real-time and (2) to compute when the camera has to be triggered for an optimal shot, taking the trigger delay into account. We implemented our approach as an Android app for smartphones. The processing starts by simply clicking on the "start" button and runs in a fully automatic manner. Numerous experiments in real-world scenarios have demonstrated the validity of the approach.

6. REFERENCES

[1] Gus Kayafas, *Stopping Time: The Photographs of Harold Edgerton*, Harry N. Abrams, 2000.

[2] Jonas Pfeil, Marc Alexa, and Carsten Gremzow, "Throwable camera array for capturing spherical panoramas," in *Diploma Thesis, TU Berlin*, 2010.

[3] Kodai Horita, Hideki Sasaki, Hideki Koike, and Kris M. Kitani, "Experiencing the ball's POV for ballistic sports," in *Proceedings of the 4th Augmented Human International Conference*, 2013.

[4] Kari Pulli, Timo Ahonen, and Alejandro Troccoli, "FCam: an architecture and API for computational cameras," in *SIGGRAPH Asia Courses*, 2011.

[5] Andrew Adams, Eino-Ville Talvala, Sung Hee Park, David E. Jacobs, Boris Ajdin, Natasha Gelfand, Jennifer Dolson, Daniel Vaquero, Jongmin Baek, Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik, Kari Pulli, Mark Horowitz, and Marc Levoy, "The Frankencamera: An experimental platform for computational photography," *ACM Transactions on Graphics (SIGGRAPH)*, 2010.

[6] Andrew Maxwell-Parish, "Automatically take perfect jump shots," `http://www.instructables.com/id/Automatically-Take-Perfect-Jump-Shots/`, accessed on 2014/02/14.

[7] Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco Paulsen, Olivier Saurer, and Marc Pollefeys, "Live metric 3D reconstruction on mobile phones," in *International Conference on Computer Vision*, 2013.

[8] Gerald M. Gregorek, "Aerodynamic drag of model rockets," Tech. Rep., Estes Industries, 1970.

[9] Jianbo Shi and Carlo Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[10] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[11] Bruce D. Lucas and Takeo Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.