Autotrigger by Example

(CS294-69 Computational Photography and Image Manipulation)

Fall 2011

Wesley Willett
Computer Science Division
University of California, Berkeley
Berkeley, CA
willettw@cs.berkeley.edu

Abstract

Triggering a camera at the right time can be difficult. When photographing a dynamic or active subject, photographers often wait for long periods of time for the subject to enter a particular pose or location in the frame. However, once the subject is appropriately framed and posed, a photographer may have only a fraction of a second to trigger the camera. We propose semi-automatic camera trigger system that allows photographers to interactively specify a *target* photo by manipulating existing images, then let the camera choose when to trigger the shot. We demonstrate several interaction techniques for manipulating input photographs and apply our approach in two example applications.

1. Introduction

Timing is important when photographing a dynamic or active subject. Photographers often wait for long periods of time for a subject to enter a particular pose or location in the frame. However, once the subject is appropriately framed and posed, a photographer may have only a fraction of a second – what Henri Cartier-Bresson termed "the decisive moment" [1952] – in which to trigger the camera.

Appropriate triggering is especially common of wildlife photography, where the subjects are difficult to direct. Wildlife photographers often wait for hours or days for a particular shot or use motion-triggered cameras to collect hundreds or thousands of arbitrary photos of a subject in the hope that one of them will capture the desired moment. Similar problems also arise in portraiture, sports photography, and other domains.

While a huge variety of computer graphics techniques exist for modifying, manipulating, and recomposing photographs, modified images are typically not acceptable in the sciences, journalism, and other fields that value the integrity of the image. Another option is to capture video, rather than photos, and then select the best frames. Unfortunately, video currently cannot match the resolutions and frame rates of still cameras. Moreover, using video makes it impossible to use flashes, requiring photographers to constantly light the entire scene - something that may not be possible in the field.

We propose semi-automatic camera trigger system that allows photographers to interactively specify a *target* photo by manipulating existing images. A set of matching algorithms then compares this target image against a low-resolution live video stream from the camera and triggers the camera to take a high-resolution still whenever a suitable shot is found. This allows photographers to specify a photo in advance and allow the camera to take the shot at the appropriate moment. We demonstrate several interaction techniques for manipulating input photographs and discuss a range of example applications.

2. Related Work

Specific variants of autotrigger are already widely available in commercial point-and-shoot cameras in the form of "smile detection" – which triggers an exposure when smiles are detected in the scene – and "blink detection" – which can delay a shot or alert the photographer if a subject's eyes are closed. However, these autotrigger mechanisms are specialized to detect specific facial features and cannot be de customized by the photographer.

Recently, a number of projects have proposed techniques for detecting relevant (and irrelevant) frames in video. Albuquerque et al. [2008] used supervised learning to train classifiers to identify good and bad expressions in video frames. More recently Fiss et al. [2011] built a predictive model of interesting expressions based on an experiment with human subjects and used it to extract "candid" portrait stills. Alternately, Bernstein et al.'s Adrenaline system [2011] used crowdsourced workers to rapidly select interesting views from video. Our approach seeks to identify pertinent still images given an input video stream, but does so in real time in order to trigger the capture of a high-resolution still image from a DSLR camera. Our approach also seeks to provide the photographer a high level of control, allowing them to specify arbitrary target images by manipulating existing imagery.

3. Approach

For the sake of prototyping, we limit discussion to photography setups where a camera is attached to a computer, which can be used to control the shot and examine captured images. This sort of "live view" shooting is popular among studio photographers who shoot products and portraits, and is often used for small-scale wildlife portraiture. Shooting using a computer allows a photographer to examine photos immediately at high-resolution in order to verify that subjects are in focus and make sure that subtle details of the shot have been captured as desired. For our purposes, shooting from a computer allows us utilize additional computing resources beyond the camera and allows photographers to use standard photo-manipulation techniques to author target images.

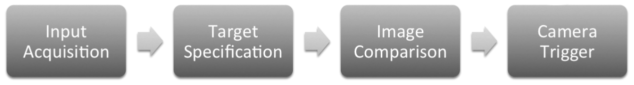

Figure 1. *Autotrigger processing pipeline.*

Our pipeline consists of four steps: First a photographer captures one or more input images of a scene. Then, using simple image manipulation techniques, the photographer selects and recombines pieces of the input images to create a target image that specifies the desired photograph. The photographer may choose to specify a complete target frame, in which the entire desired image is specified, or a partial target, in which only some portion of the desired photograph is defined. Next, our system compares the target image against a live, low-resolution video stream from the camera, looking for video frames that match the target image. Finally, when a suitable match is found, the system triggers the camera, capturing a high-resolution still image.

3.1. Target Specification

Our target specification interface allows photographers to capture or upload one or more images and then manipulate those images to create a target image that specifies when the camera should be triggered. We support several different interactions for specifying target images.

Weight-painting

In the simplest mode, a photographer can specify constraints by painting weights onto a single input image (Figure 2, left). A photographer can paint positive weights onto regions of the image that should match the original image and negative weights on portions of the image that should be different. For example, if the specimen in a wildlife portrait blinks, marring an otherwise ideal photo, the photographer could paint negative weights onto the subject's eyes, indicating that the camera should attempt to capture an alternate image that is identical but with different eyes. Alternately, the photographer could paint positive weights a subject's body, indicating that any photo with the subject's body in that position would be acceptable. Weights can be painted onto the image at varying strengths, allowing a photographer to place a stronger requirement on some specified regions than others.

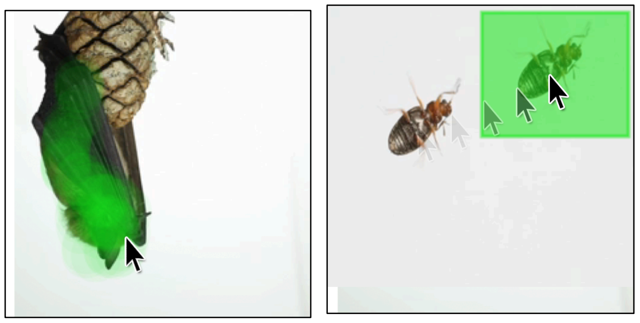

Figure 2. *Weight painting (left) and marquee manipulation (right). Positive weights are indicated in green.*

Marquee-manipulation

In some cases a photographer may wish to reposition elements within the target image. We support this via a set of marquee selection and manipulation tools, that allow the photographer to drag a selection on the original image and reposition it elsewhere in the target image (Figure 2, right). These repositioned portions of the image are automatically weighted.

Outside Image-editing Tools

If photographers wish to perform additional, more complicated edits on the target image, they can export the target and make edits externally using more powerful image editing tool like Adobe Photoshop. Photographers can then re-import the edited image along with custom weights.

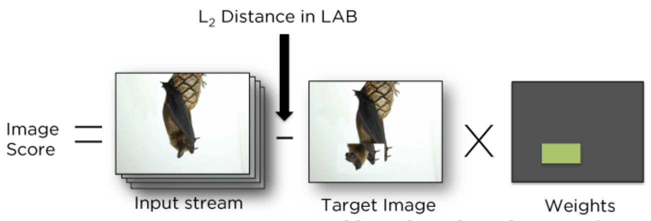

Figure 3. *Input images are scored based on their distance from the target image in the weighted regions.*

3.2. Image Comparison

In order to determine whether or not a frame of input video is a good match for the target image, we compute a distance score that quantifies the difference between the input frame and the target (Figure 3). To compute this score we perform pixel-level comparisons between images, sampling based on the weights applied to the various image regions. In all cases, we compute the distance between pixel values using the L2 distance in the LAB color space. We sum these distances and normalize them based on the number of pixels sampled and the weights applied to them to produce a score for each video frame.

Sampling

We observe that photographers may often wish to constrain a few regions of the target image. Based on this observation we implemented a weight-based sampling scheme for comparing images, sampling only at locations where the photographer has applied weights to the target image. The system can also adjust the sampling density based on weighting, sampling more frequently in heavily weighted areas and less frequently in areas where a photographer has applied lighter weighting, then normalizing the results.

Because the target images and input video in our prototype are both low-resolution (320x240 pixels), the cost of computing the difference on all pixels in the image is low, and sampling is not strictly necessary (we typically sample all pixels). However, our sampling scheme should allow the same comparison metrics to work on larger input and target images or on slower hardware while still ensuring real-time performance.

Jitter

When a photographer specifies a constraint in a particular region of an image, they typically do not desire a pixel-level match. A small amount of spatial deviation is typically acceptable, and can dramatically increase the odds of finding a suitable shot. To address this, we allow photographers to relax the spatial constraints on matches, allowing matches within a specified radius of the original point. For each point in the target image we sample multiple nearby points in the input video frame using a Gaussian distribution based on the user-specified radius. We compare each point from this distribution against the original point from the target image and use the smallest distance when computing the total difference between images.

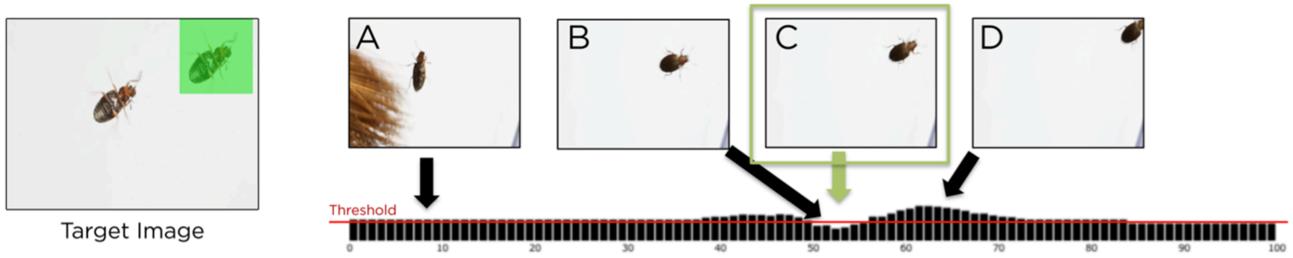

Figure 4. *A target image (left) and four still images from a video. The timeline below the thumbnails shows the relative scores of each frame in the video clip. Note that while frame B scores below the threshold, frame C is actually the best choice.*

Downsampling
Target images and video frames may be noisy, causing pixel-level comparisons to yield poor results, even for very similar images. To help reduce sensitivity to noise and small-scale variations between images, we downsample both our target and input images before performing the comparison. We default to downsampling our low resolution video (320x240 pixels) by a factor of 10, producing 32x24-pixel images. This downsampling threshold is user-controllable and photographers can manipulate it to vary sensitivity to detail.

Thresholding
Choosing an appropriate scoring threshold at which to trigger the camera is difficult. Because future frames are always unknown, it is always possible that a future frame will be a better match than the current one (unless the target and input images are perfectly identical). For example, in Figure 4, the best match occurs at point C and, ideally, we would trigger the camera then to capture the best shot. Unfortunately, at point B, we do not know yet whether or not the current frame is a local minimum, or if a future frame will be an even better match.

Our current system provides a photographer with a live graph showing the scores of the past 100 video frames. Using this graph, the photographer can set a threshold at which the camera will be triggered. The system then monitors incoming frames and, once the score dips below the photographer's threshold, triggers the camera the next time the score increases. This prevents the camera from triggering early (e.g. at point B), but may still miss the true minimum. A simple predictive model that introduces some smoothing on the input scores should produce better results, but we leave this for future work.

3.3. Implementation
We constructed our prototype by connecting a Digital SLR Camera (Canon T3i) to a laptop. Because accessing a live video stream from the camera was non-trivial, we instead used live video from a small webcam mounted directly above the camera's lens (Figure 5). The target specification interface and image comparison code were implemented as web applications using HTML5 and JavaScript. We used a Sikuli script [Yeh 2009] to trigger the camera by programmatically depressing the shutter button in an instance of Canon's EOS Utilities application running on the laptop.

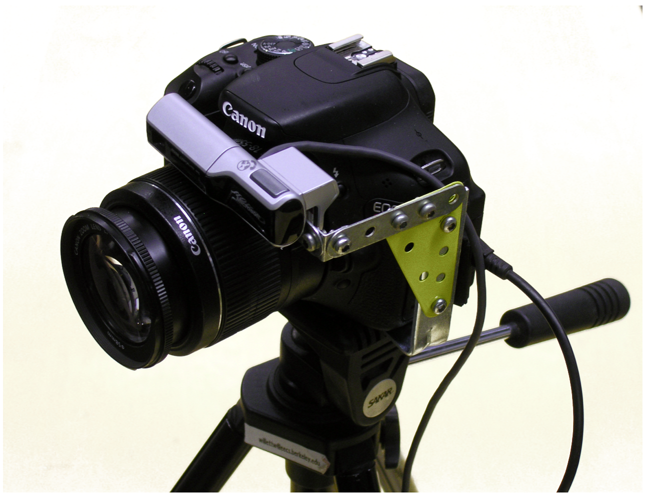

Figure 5. *DSLR with attached webcam.*

4. Applications
We have explored two applications of our technique.

Automated Camera Traps for Wildlife Photography
The most typical application of our system is for camera traps and wildlife portraiture (Figures 2-4). Here, a photographer can capture a background photo of the scene for context, then either coax the specimen into the scene (which may be possible in a

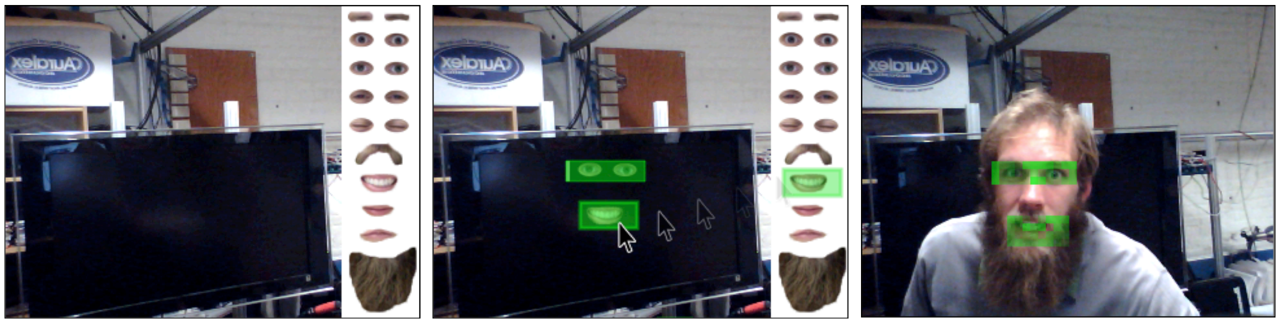

Figure 6. *In the Portrait Stickers mode, a photographer captures a reference image of a scene (left) and then drags facial features into the scene to create a target image (center). The image at right shows the best (terrifying) image (with weights) from a short photo shoot.*

studio setup) or introduce stock photos of the desired specimen (a more likely proposition in the field). The photographer can then manipulate the image and judiciously apply weights in order to build a targeted camera trap.

Portrait Stickers

Photographers can also use existing templates to specify desired shots. In the *portrait stickers* mode (Figure 6), the interface provides various sets of eyes, mouths, and other facial features that can be dragged into the target image in order to specify desired facial expressions. These can be added to a blank canvas to trigger the camera when faces occur, or added on top of captured images to specify a small deviation (e.g. open eyes or a bigger smile) on an existing portrait.

4.1. Failure Cases

During our initial development, we identified a number of failure cases for our current algorithm. These include:

Changes in Size and Orientation

While we jitter our samples to help compensate for spatial variation in X and Y, our system cannot currently handle cases in which the subject is slightly closer or further from the camera or is rotated in X and Y. Introducing size and rotation jitter in our sampling algorithm could help compensate for this.

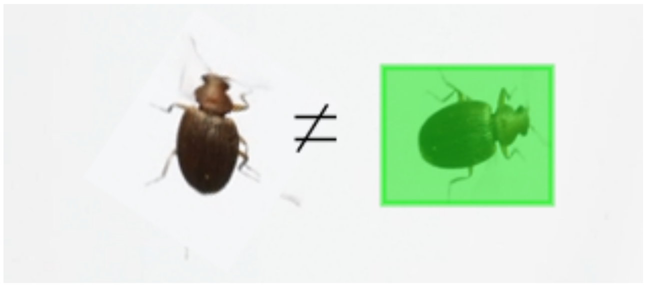

Figure 7. *Patches in the target image are not robust to changes in size or rotation. The rotated insect (left) will not match the target specified on the right.*

Lighting Changes

Our current scoring algorithm depends on L2 distance in the LAB color space and is not robust to lighting changes in the environment. Removing the luminance term from the distance method should improve performance in under varying light, but may still struggle if the color or direction of the lighting changes. A scoring system that compared edge orientations or another shading-independent attribute might provide even better results.

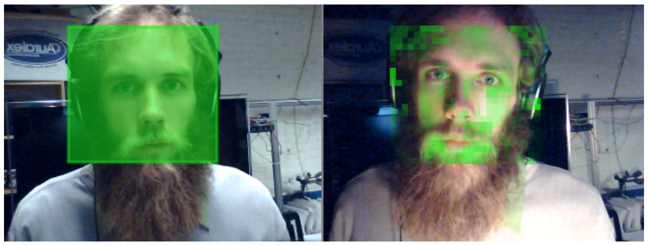

Figure 8. *Patches in the target image (left) are not robust to changes in lighting color and direction. An input video frame (right) shows large errors when the lighting direction changes from the upper right to the lower right.*

Over-constrained Target Images

In practice, we find that it is often easy to apply too many constraints to a target image, ensuring that a matching frame is unlikely to occur. For example, using the marquee to select a subject's head typically also results in selecting some portion of the background. These background artifacts may unintentionally penalize good input frames, causing them to score poorly.

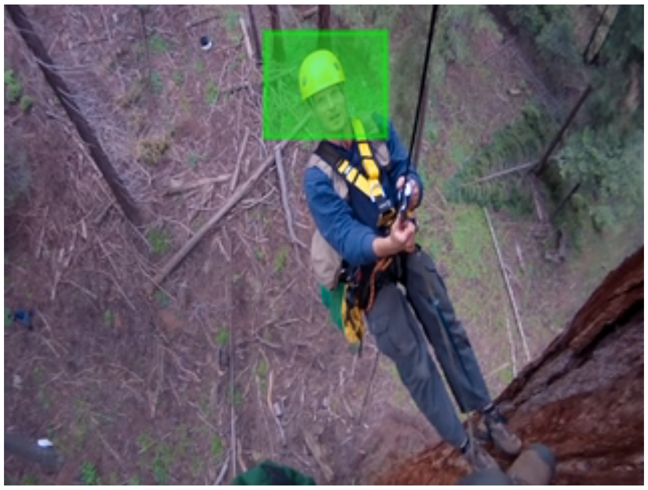

Figure 10. *Selecting a rectangular region in the image can unintentionally force input frames to match the background included in the marquee region, even if only the helmet in the foreground was intended.*

4.2. Performance

Our current implementation suffers from relatively long lag times (Figure 11), making it difficult to trigger the camera at the appropriate time. A typical cycle of the complete system takes approximately 1.75 seconds. Of this, about half a second is required to extract a frame from live video, while only about 10-20 milliseconds are required to compute a difference score between the new image and the target. Once the decision to trigger the camera is made, there is a roughly one second delay during which the Sikuli script recognizes a successful match and depresses the virtual trigger. An additional 0.25-second delay occurs between the time when the virtual button is depressed in the Canon EOS Utility and the time when the camera shutter finally opens.

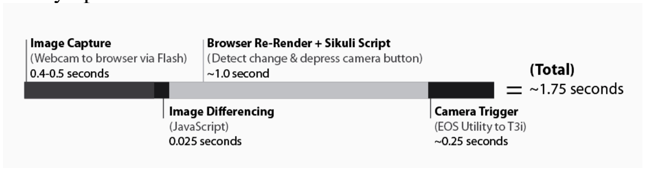

Figure 11. *Timing for a single camera trigger.*

The actual computation time - even when sampling every pixel in our 320x240 pixel images - is very small and can easily be computed at frame rates faster than the camera can deliver (>30 fps). Reading a new image from an existing video buffer in the browser (rather than from the webcam) is also extremely fast. This suggests that the entire pipeline could be handled in real time given a more responsive mechanism for reading video and triggering the camera. One possible solution is to utilize an outside library like *libgphoto* (www.gphoto.org/proj/libgphoto2/). that provides low-level access to the underlying camera ports and protocols. However, at this time, no such libraries appear to support the DSLR we used.

5. Future work

Our prototype system represents a first step into the space of semi-automated automatic camera triggers, and suggests a range of further extensions.

Translation to Camera

While we prototyped our system using a webcam attached to a DSLR, this solution has a number of drawbacks. Using webcam video means that the input video often has a field of view that differs substantially from that of the camera (particularly when zooming). Most webcams also attempt to adjust the exposure and white balance of their subjects dynamically and can introduce distortions and color variations that make it difficult to compare images taken at different times. Future versions of the system should use the live view video stream directly from the DSLR.

Ideally, in future work, our entire pipeline could be moved onto the camera, allowing photographers to capture and manipulate target images without the need for a PC. In fact, the set of image manipulation operations we included in our prototype are all reproducible on a small, touch-screen display. This means that a similar autotrigger system could readily be implemented on a modern smartphone and likely on future camera hardware.

Manipulations

We implement a very limited set of image manipulation operations in our target specification interface. The current marquee-drawing and manipulation tools, in particular, restrict the expressivity of the tool and make many desirable target images difficult to compose. Because photographers can only select rectangular regions, it is often impossible to select a foreground image without also selecting the background around it. Adding a lasso selection tool or allowing photographers to select foreground subjects using GrabCut [Rother 2004] would make repositioning specific subjects much easier.

The current system also offers no way to scale or rotate selected regions of the image. Adding standard transformation and rotation handles to selected regions would allow this kind of manipulation. A more complete system might also allow photographers to interactively specify the amount of position, rotation, and size jitter permitted in a match. This would give photographers more freedom to specify target images that match a wider range of shots.

Additional Constraints

We allow photographers to apply only image-based constraints to specify target images, however, in some cases it may also be useful to apply constraints based on image statistics or camera properties. For example, a photographer may wish to take photos only when a scene is sufficiently illuminated, when the noise level is low, or when a specific region of the image is in focus.

6. Conclusion

We have demonstrated a semi-automatic camera trigger system that allows photographers to interactively specify a target photo by manipulating existing images. Our system introduces a set of interaction techniques that allow photographers to build target images by manipulating and weighting reference images. A set of matching algorithms then compare the target image against a low-resolution live video stream from the camera and trigger the camera whenever a suitable shot is found. This allows photographers to specify a photo in advance, then let the camera to take the shot at the appropriate time.

References

1. ALBUQUERQUE, G., STICH, T., SELLENT, A., AND MAGNOR, M. 2008. The good, the bad and the ugly: Attractive portraits from video sequences. *In Proc. 5th European Conference on Visual Media Production (CVMP 2008).*

2. CARSTEN ROTHER, VLADIMIR KOLMOGOROV, AND ANDREW BLAKE. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers* (SIGGRAPH '04), Joe Marks (Ed.). ACM, New York, NY, USA, 309-314.

3. CARTIER-BRESSON, H. 1952. *The Decisive Moment*. Simon & Schuster.

4. JULIET FISS, ASEEM AGARWALA, AND BRIAN CURLESS. 2011. Candid portrait selection from video. In*Proceedings of the 2011 SIGGRAPH Asia Conference* (SA '11). ACM, New York, NY, USA, , Article 128 , 8 pages.

5. MICHAEL S. BERNSTEIN, JOEL BRANDT, ROBERT C. MILLER, AND DAVID R. KARGER. 2011. Crowds in two seconds: enabling realtime crowd-powered interfaces. *In Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 33-42.

6. TOM YEH, TSUNG-HSIANG CHANG, AND ROBERT C. MILLER. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 183-192.