Client-side Backprojection of Presentation Slides into Educational Video

Yekaterina Kharitonova[†*], Qiyam Tung[†], Alexander Danehy[†], Alon Efrat[†], Kobus Barnard[‡†*]
U. of Arizona: [†]Department of Computer Science, [‡]School of Information: Science, Technology and Arts;
[*]The iPlant Collaborative
{ykk, qwvako, adanehy, alon, kobus}@email.arizona.edu

ABSTRACT

A significant part of many videos of lectures is presentation slides that occupy much of the field of view. Further, for a student studying the lecture, having the slides sharply displayed is especially important, compared with the speaker, background, and audience. However, even if the original capture supports it, the bandwidth required for real time viewing is substantive, especially in the context of mobile devices. Here we propose reconstructing the video on the client side by backprojecting high resolution slide images into the video stream with the slide area blacked out. The high resolution slide deck can be sent once, and inserted into the video on the client side based on the transformation (a homography) computed in advance. We further introduce the idea that needed homography transformations can be approximated using affine transformations, which allows it to be done using built-in capabilities of HTML 5. We find that it is possible to significantly reduce bandwidth by compressing the modified video, while improving the slide area quality, but leaving the non-slide area roughly the same.

Categories and Subject Descriptors

K.3 [**Computing Millers**]: Computers and Education

Keywords

presentation slides, lecture video, homography, affine transformation, backprojection

1. INTRODUCTION

Mobile devices are becoming one of the most common ways to access the Web. They are also being utilized as educational tools as part of the growing field of mobile learning. Videos of lectures and talks are becoming more prevalent as more universities and companies record or stream the presentations and make the videos available online ([1], [8], [6], [3], [4]). In a typical presentation the camera operator does not change the camera settings very often. The operator usually makes the projected slide fill much of the video frame. Quite often the presentations are captured using a relatively inexpensive camera that produces a low quality recording. Even if a presentation was recorded in a high-definition format, the video is typically post-processed and compressed for reasonable on-line access. The post-processing typically reduces the video bit-rate, compressing it, and making it blurry even if it was not already so. The poor video quality makes it hard to read the content of the presentation slides or to discern which slide is being shown.

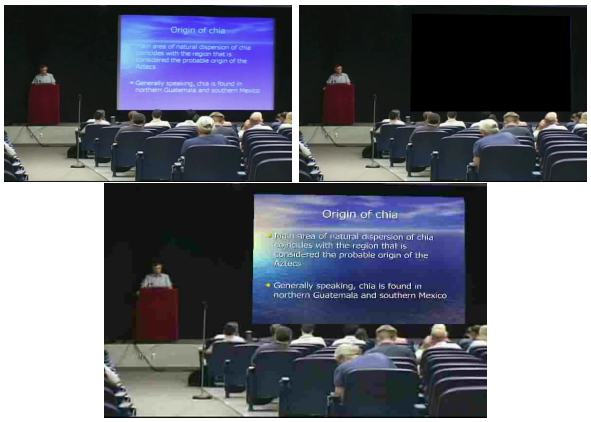

Figure 1: Overview of the method. Top left shows the original video, top right shows the slide removed and replaced with black pixels. The bottom shows the slide backprojected over the slide area. Notice that it is not crucial that the non-black area in the top right image is crisp, but that making the slide area crisp is important for educational users.

We propose a solution that considers the slide area and the region outside it separately. We modify the video so that the slide area is blacked out allowing further compression. The slide deck is transfered in a high resolution format, and inserted into the video stream using appropriate transformations.

Several systems have been developed where slides are shown side-by-side with the video stream ([5], [7], [9]). However, such systems are cumbersome to use on mobile devices with limited screen resolution. Providing the video with a clearly visible backprojected slide region eliminates the need to use additional space on the mobile screen to show the high-resolution slide image, as considered by others. For example, Tung et al. [16] backproject presentation bullet points for

MM'12, October 29–November 2, 2012, Nara, Japan.

viewing on mobile devices. The advantage of our method is that it doesn't require server-side backprojection of the entire video, reducing the amount of data sent to the client. Performing backprojection on the client-side, thus produces the best possible results in terms of video quality and bandwidth conservation.

2. BACKPROJECTION

Our approach relies on the accurate geometric transformations (*homographies*) between the slides and the frames. The process of projecting the images of slides back into the corresponding video frames is referred to as *backprojection*.

Substituting high-resolution images of slides into a video produces sharper-looking slide content than what originally appeared in the video. Given the homogeneous slide points, $\mathbf{s} = [x, y, w]^T$, they are projected into the points on a frame, $\mathbf{p} = [u, v, w']^T$, by a homography $\mathcal{H}$:

$$\mathbf{p} = \mathcal{H}\mathbf{s} \tag{1}$$

We use this correspondence to link the frame pixels to the areas of the slide that they map to. As shown in Figure 2, a single pixel of a frame can map to an area of a slide consisting of multiple high resolution pixels. We use the approach of Winslow et al. [17], to backproject the pixels from the frame to the corresponding slide regions. This method uses the iterative scanline algorithm to map every pixel of a video frame to the slide area and then determines the average of the corresponding slide pixels. Automated matching of the slides to video frames and the accurate computation of $\mathcal{H}$ has been developed by Fan et al. and others [10–15].

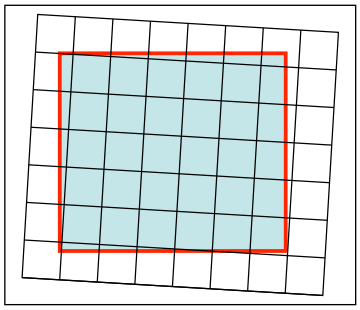

Figure 2: An illustration of a single frame pixel (the axis-aligned red box) maps onto a larger area of the slide image, which consists of full and partial pixels (represented by the shaded grid).

We observed that while backprojection requires accurate homographies for all frames, most of the transformations in a typical video can be approximated using *affine* transformations. In a typical presentation, a camera operator may occasionally zoom-in on the slide, zoom-out to show the speaker and the audience, and pan the camera. These events do not require applying the full homography to all frames – instead, the transformations may be approximated by scaling and translating the first frame image in the event sequence, which has been backprojected using an appropriate homography. The recent advancements in the HTML technologies provide the ability to apply these transformations directly on the client-side.

3. HTML5 <CANVAS> ELEMENT

The HTML5 `<canvas>` element is a drawable bitmap region displayed on a web page. It can be used to dynamically draw and script graphics using JavaScript. It has several built-in methods for adding and manipulating images. `canvas` also provides a native integration with the HTML `<video>` tag. Thus, it gives us the ability to manipulate a backprojected slide image independently from the video frame. By combining the two, we can overlay images of slides onto the video frames directly in the client's browser.

While it can be desirable to execute the complete homography transformation on the client-side, the `canvas` element does not natively support doing so, and thus this "direct" method is computationally expensive. However, the `canvas` provides a built-in `transform()` method, which supports translation, scale and rotation, possibly taking advantage of graphics hardware.

For video frames where the affine transformation is too erroneous, we reinitialize by drawing a server-side computed homography-backprojected image of a slide on a transparent background. We then manipulate it using the computed affine transformations to achieve the desired backprojection effect. Thus, instead of backprojecting a video on the server-side, we only send the client a small set of images, which can be transformed on-demand right in the browser. If, however, the mobile device has a powerful CPU, then it is still possible to perform a direct pixel-wise manipulation in the `canvas`, achieving the true backprojection on the client-side.

4. HOMOGRAPHY APPROXIMATION

Our goal is to approximate the transformation of a slide S throughout the sequence of frames $\{F^j, ..., F^n\}$. Let's define the four points corresponding to the slide S's corners as $s_k = \{s_1, s_2, s_3, s_4\}$. Their homogeneous image coordinates are: $s_1 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$, $s_2 = \begin{bmatrix}S_{width} & 0 & 1\end{bmatrix}^T$, $s_3 = \begin{bmatrix}S_{width} & S_{height} & 1\end{bmatrix}^T$, $s_4 = \begin{bmatrix}S_{height} & 0 & 1\end{bmatrix}^T$.

Let $\mathcal{H}^a$ be a homography transformation for frame F^a. Applying $\mathcal{H}^a$ to the corner points, s_k, produces a set of four points projected into the frame, F^a (see Figure 3).

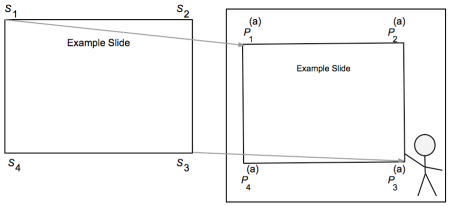

Figure 3: Projecting the corners of slide S (left) into the video frame F^a (right).

These points can be represented by a 3x4 matrix $P^{(a)} = \begin{bmatrix}p_1^{(a)} & p_2^{(a)} & p_3^{(a)} & p_4^{(a)}\end{bmatrix}$, where each column is given by $p_k^{(a)} = \begin{bmatrix}u_k^{(a)} & v_k^{(a)} & 1\end{bmatrix}^T$. These homogeneous coordinates were obtained from $\mathcal{H}^a s_k = \begin{bmatrix}x_k^{(a)} & y_k^{(a)} & w_k^{(a)}\end{bmatrix}^T$, scaled by $w_k^{(a)}$ (where $k = 1, ..., 4$). Because we calculated the true homographies, we can compute these projected coordinates, $P^{(a)}$, for all frames.

Consider two frames F^a and F^b. We can approximate the points from F^b by applying an affine transformation, T^b, to the points of frame F^a. Such transformation can be represented by a homogeneous matrix T^b:

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

which, since the last row is fixed, we can express by the two row vectors $t_1 = \begin{bmatrix} t_{11} & t_{12} & t_{13} \end{bmatrix}$ and $t_2 = \begin{bmatrix} t_{21} & t_{22} & t_{23} \end{bmatrix}$, so that $T = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$.

Let $q_k^{(b)} = \begin{bmatrix} x_k^{(b)} \\ y_k^{(b)} \end{bmatrix}$ be the image points on frame F^b that were approximated by applying T^b to the corresponding image coordinates, $p_k^{(a)}$, from the previous frame, F^a.

$$p_k^{(b)} \approx q_k^{(b)} = T^b p_k^{(a)} \quad (3)$$

Then, for each projected corner point, $q_k^{(b)}$, we get the following coordinates

$$x_k^{(b)} = t_1 p_k^{(a)} = t_{11} u_k^{(a)} + t_{12} v_k^{(a)} + t_{13} \quad (4)$$

$$y_k^{(b)} = t_2 p_k^{(a)} = t_{21} u_k^{(a)} + t_{22} v_k^{(a)} + t_{23} \quad (5)$$

We want to find such T^b that would minimize the distance between the true projected coordinates $p_k^{(b)}$ and the unknown points $q_k^{(b)}$.

If we re-write T as a column vector $t = \begin{bmatrix} t_1^T \\ t_2^T \end{bmatrix}$, we can compute the coordinates for all four approximated projected points $q_k^{(b)}$ by arranging the coordinates of the points $p_k^{(a)}$ into a new matrix, U:

$$p^{(b)} \approx q^{(b)} = \begin{bmatrix} x_1^{(b)} \\ y_1^{(b)} \\ x_2^{(b)} \\ y_2^{(b)} \\ x_3^{(b)} \\ y_3^{(b)} \\ x_4^{(b)} \\ y_4^{(b)} \end{bmatrix} = \begin{bmatrix} u_1^{(a)} & v_1^{(a)} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1^{(a)} & v_1^{(a)} & 1 \\ u_2^{(a)} & v_2^{(a)} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2^{(a)} & v_2^{(a)} & 1 \\ u_3^{(a)} & v_3^{(a)} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3^{(a)} & v_3^{(a)} & 1 \\ u_4^{(a)} & v_4^{(a)} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_4^{(a)} & v_4^{(a)} & 1 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \end{bmatrix} \quad (6)$$

Thus, $q^{(b)} = Ut$, where $q^{(b)}$ is an 8x1 column vector with the projected x and y coordinates for each of the four corner points.

Now, we can find such t (corresponding to T^b) that would minimize the error between the true projected coordinates $p_k^{(b)}$ and the estimated coordinates $q_k^{(b)}$. Using the linear least squares, we solve for t that would minimize the squared error, $E = |e|^2 = e^T e$, where

$$e = Ut - p_k^{(b)} \quad (7)$$

The solution to the above equation is

$$t = U^\dagger p_k^{(b)} \quad (8)$$

where $U^\dagger = (U^T U)^{-1} U^T$ is the pseudoinverse of U.

We can now estimate the position of the slides in the video frames using the linear transformations encoded by t.

	$\tau = 2$	**Size**	$\tau = 3$	**Size**
# frames	3637	127 Mb	2124	74 Mb
# affine matrices	2175	15 Mb	3688	8.6 Mb

Table 1: The number of backprojected frames to be sent to the client based on the pixel coordinate distance threshold τ in a test video. The affine homographies are computed between every other frame. This test video had 117 slide transitions and a total of 71387 frames.

5. SYSTEM IMPLEMENTATION

Geometric Transformations. Since we are projecting slides onto a video, we need accurate frame-to-slide correspondences and the homographies between all frame-slide pairs. Using these homographies we compute the coordinates of the slide corners in each frame. Then, using the method described in the previous section, we estimate the affine transformation between the successive frames. Finally, we backproject the corners using the computed affine transformations and compare these new projected coordinates, **q**, to the ones obtained using the homographies, **p**. If the average re-projection error (in pixels) of the approximated points is within the threshold τ of the homography-transformed points, then we use an affine transformation for that frame. Otherwise, we use that frame's homography to project the corresponding slide onto a transparent image. The greater the threshold τ, the further we let the approximated points drift from the homography-transformed points, and the fewer images we have to send to the client. Note that for each slide change in the video, we need to send a backprojected image.

Since our sample presentation videos had no camera movement, we determined an optimal value of τ by analyzing a short video featuring many camera transitions. The results for different values of the threshold are shown in Table 1.

Video Backprojection. To verify that our modifications can reduce bandwidth, we created two types of videos for comparison. One has backprojected slides and the other has the slide area blacked out. To do this, we first extract all video frames, backproject each with the appropriate slide (or black slide), and then reassemble the video.

We first compress the videos under the same settings as the original video (600 kbps), using the Theora video codec as it is compatible with HTML5. The videos are compressed using the program `ffmpeg2theora` [2] with variable bitrate (VBR). We choose to use VBR so that `ffmpeg2theora` could reduce the bitrate for scenes whenever possible. We then experimented with additional average video bitrates for the video with black slides to reduce the size without significantly degrading the visual quality. Table 3 shows the space savings resulting from using different compression settings.

6. EXPERIMENTS AND EVALUATION

To evaluate the effectiveness of the proposed method, we used videos with ground-truth frame-to-slide matches (*segmentation*) as well as the location of slide corners in the video frames.

We compared the visual quality of the video backprojected with the black slide against the original, non-backprojected video. We found that we can lower the compression rate

Video ID	WC			FB			LA		
Compression Setting (kbps)	**600**	**400**	**200**	**600**	**400**	**200**	**600**	**400**	**200**
Backprojected Video	171.9	129.1	52.7	227.8	159.4	90.4	227.0	165.2	95.2
Modified Video	102.9	75.1	47.1	135.4	107.6	73.5	117.8	94.1	70.3
Video Saving	40.1%	41.8%	10.6%	40.6%	32.5%	18.7%	48.1%	43%	26.2%
Overall Saving	39.1%	40.5%	7.4%	40.0%	31.7%	17.4%	46.5%	40.9%	22.4%

Table 3: The sizes (in Mb) of the three backprojected videos and the modified videos, which were created by backprojecting a black slide. The overall savings take into account the slide image data.

ID	Number of slides	Slide deck size	Number of frames	Video size
WC	48	1.7 Mb	71387	187.6 Mb
FB	37	1.2 Mb	87784	229.4 Mb
LA	103	3.6 Mb	91770	240.5 Mb

Table 2: Presentation data statistics for the three videos. The size of the high-resolution slide deck is small compared to that of the video.

down to 400 kbps and the area outside of the slide would still look very close to how it appears in the original video. Table 3 shows that for all videos the size of the resulting compressed video was much smaller than the fully backprojected video.

If the client has sufficient CPU resources, direct homography computation can be done using the `canvas`. With this method, the bandwidth requirements consist of the modified video, the slide images (small, see Table 2), and the homography data (negligible). The overall savings, as a function of the compression settings, are shown in Table 3.

If the client cannot perform computationally-intensive operations, then we fall back on applying the affine method with `canvas`. The bandwidth needed beyond that for the direct method is due to any extra slide images needed to keep the average reprojection error less than τ pixels using affine transformations. Thus the extra bandwidth needed depends on how much camera change there is, and the mapping error threshold τ (see Table 1).

The three videos in Table 3 have little camera change, and thus the savings using the affine approximation method are similar to those using direct backprojection. Note that the affine method uses less CPU which may be significant for some applications.

We have run our algorithm on one of the test videos, which had 87784 frames. For $\tau = 3$, the resulting bandwidth cost consisted of 107.6 Mb modified video (400 kbps), 38 Mb for 1189 affine transformation files and about 41 Mb for the homography-transformed images. That's 18% savings, given that the corresponding backprojected video file was 227.8 Mb.

7. ACKNOWLEDGEMENT

Resources from the iPlant Collaborative contributed to the research results reported in this paper. The iPlant Collaborative is funded by a grant from the National Science Foundation (# DBI-0735191).
URL: `www.iplantcollaborative.org`

8. REFERENCES

[1] Academic Earth. `http://academicearth.org/`.
[2] ffmpeg2theora. `http://v2v.cc/~j/ffmpeg2theora/`.
[3] Free Online MIT Course Materials. `http://ocw.mit.edu/OcwWeb/web/courses/av/`.
[4] Open Yale Courses. `http://oyc.yale.edu/courselist`.
[5] Semantically Linked Instructional Content (SLIC). `http://slic.arizona.edu/`.
[6] UC Berkeley Webcasts. `http://webcast.berkeley.edu/courses.php`.
[7] Video Lectures. `http://videolectures.net/`.
[8] YouTube EDU. `http://www.youtube.com/edu`.
[9] E. Altman, Y. Chen, and W. C. Low. Semantic exploration of lecture videos. *Proceedings of the tenth ACM international conference on Multimedia - MULTIMEDIA '02*, page 416, 2002.
[10] N.-M. Cheung, D. Chen, V. Chandrasekhar, S. S. Tsai, G. Takacs, S. A. Halawa, and B. Girod. Restoration of out-of-focus lecture video by automatic slide matching. In *Proceedings of the international conference on Multimedia*, MM '10, pages 899–902, New York, NY, USA, 2010. ACM.
[11] Q. Fan, A. Amir, K. Barnard, R. Swaminathan, and A. Efrat. Temporal modeling of slide change in presentation videos. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.
[12] Q. Fan, K. Barnard, A. Amir, and A. Efrat. Accurate alignment of presentation slides with educational video. In *IEEE International Conference on Multimedia & Expo (ICME)*, 2009.
[13] Q. Fan, K. Barnard, A. Amir, and A. Efrat. Robust spatiotemporal matching of electronic slides to presentation videos. *IEEE Transactions on Image Processing*, 20(8):2315–2328, 2011.
[14] Q. Fan, K. Barnard, A. Amir, A. Efrat, and M. Lin. Matching slides to presentation videos using sift and scene background matching. In *8th ACM SIGMM International Workshop on Multimedia Information Retrieval*, 2006.
[15] G. Gigonzac, F. Pitie, and A. Kokaram. Electronic slide matching and enhancement of a lecture video. *IET Conference Publications*, 2007(CP534):9–9, 2007.
[16] Q. Tung, R. Swaminathan, A. Efrat, and K. Barnard. Expanding the point: automatic enlargement of presentation video elements. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 961–964, New York, NY, USA, 2011. ACM.
[17] A. Winslow, Q. Tung, Q. Fan, J. Torkkola, R. Swaminathan, K. Barnard, A. Amir, A. Efrat, and C. Gniady. Studying on the move: enriched presentation video for mobile devices. In *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, INFOCOM'09, pages 224–229, Piscataway, NJ, USA, 2009. IEEE Press.