Client-side backprojection of presentation slides into educational video

Yekaterina Kharitonova†*, Qiyam Tung†, Alexander Danehy†, Alon Efrat†, Kobus Barnard‡†*

University of Arizona: †Department of Computer Science, ‡School of Information: Science, Technology and Arts; *The iPlant Collaborative

Introduction

Motivation:

The slide area in blurry, low-quality or compressed videos is hard to read. Displaying clear and sharp slides is more informative than showing the speaker, background, and audience with the same level of detail.

Idea:

Blacking-out the slide area in the video results in smaller-sized video files, thus, reducing bandwidth. HTML5 technology allows to reconstruct the video using slide-to-frame *homographies*.

Goal:

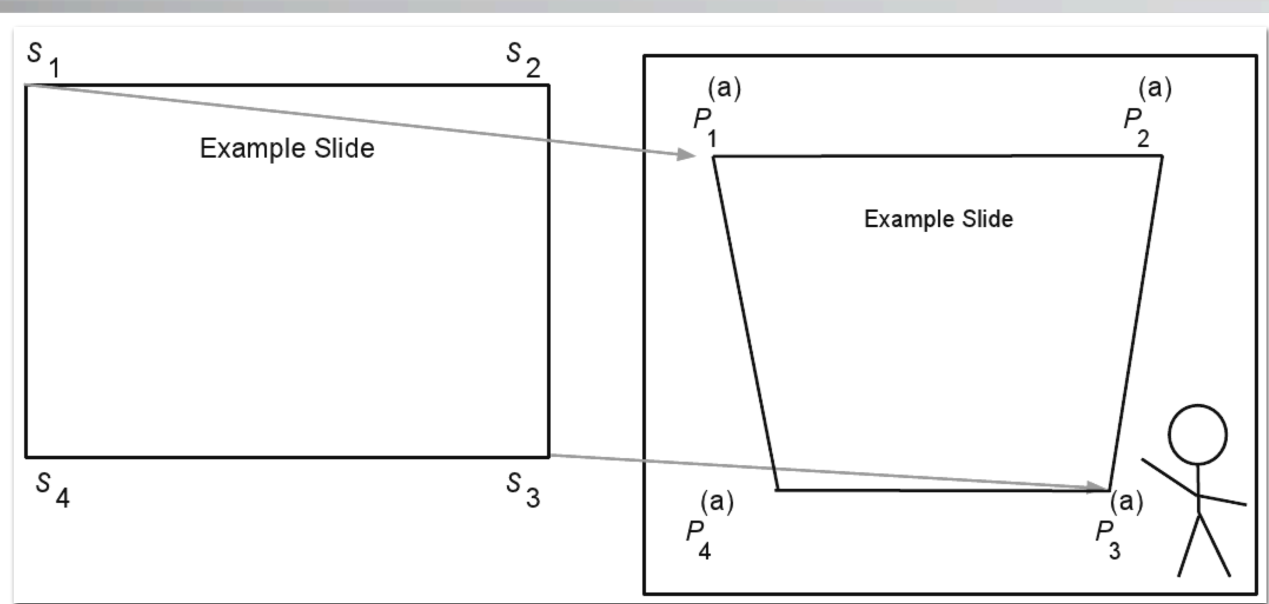
Backproject high-resolution slide images into the video stream on the client side.

Backprojection

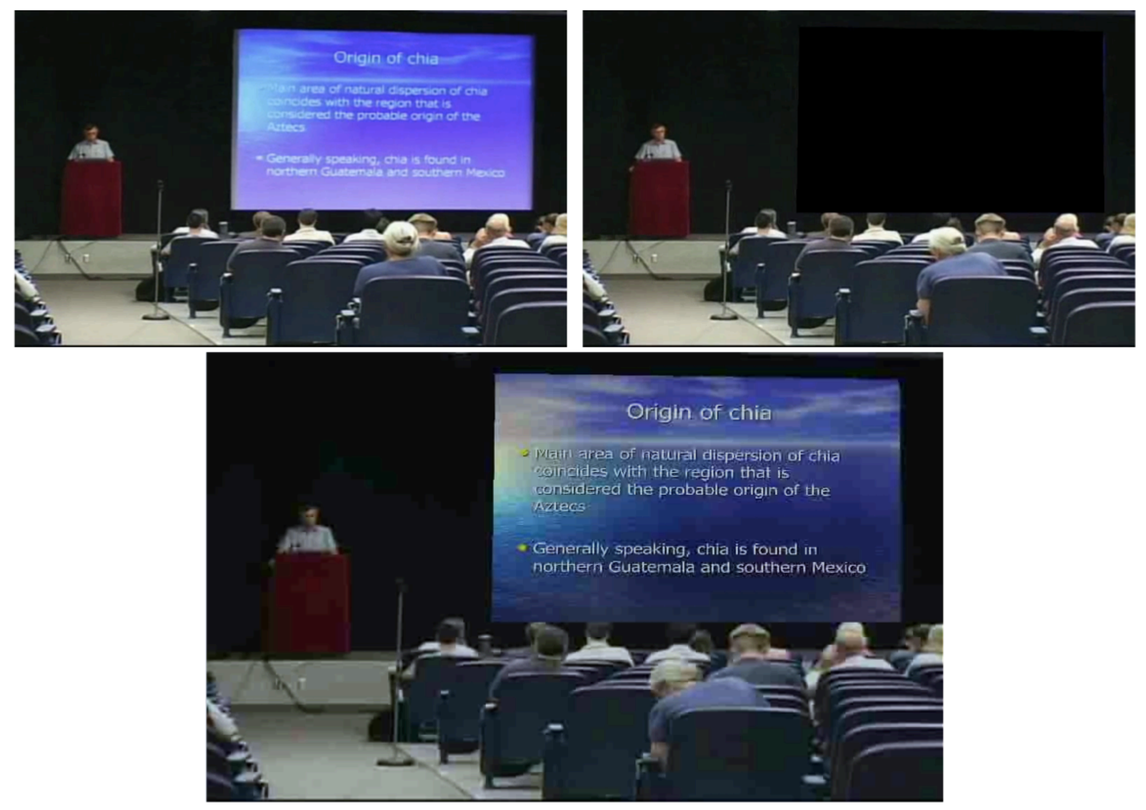
We transform homogeneous slide points, $\mathbf{s} = [x, y, w]^T$, into the frame coordinates , $\mathbf{p} = [u, v, w']^T$, by applying a homography $\boldsymbol{H}$: $\mathbf{p} = \boldsymbol{H}\,\mathbf{s}$

We can approximate $\boldsymbol{H}$ using an *affine* transformation: the camera events such as zooming-in / -out, and panning may be approximated by scaling and translating the first frame in the event sequence.

Given the two consecutive frames F^a and F^b, we can approximate points from F^b by using a homogeneous matrix $\boldsymbol{T}$, instead of the full homography, $\boldsymbol{H}$.



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



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.

Homography approximation

Let $\mathbf{q}^b$ be the image points on frame F^b, approximated by applying the affine transformation matrix $\boldsymbol{T}$ to the coordinates from the previous frame, $\mathbf{p}^a$:

$\mathbf{p}^b \approx \mathbf{q}^b = \boldsymbol{T}\,\mathbf{p}^a$

To compute the homogeneous matrix $\boldsymbol{T}$:

1. Rewrite the matrix $\boldsymbol{T}$ as a column vector $t = \begin{bmatrix} t_1^T \\ t_2^T \end{bmatrix}$
2. Place the coordinates of $\mathbf{p}^a$ into a matrix $\boldsymbol{U}$.
3. Arrange $\mathbf{q}^b$ as an 8x1 column vector, where $q^b = Ut$
4. Use the *linear least squares* to solve for $\boldsymbol{t}$ by minimizing the squared error, $E = \left|e^2\right| = e^T e$, where

$e = Ut - p^b \rightarrow t = U^* p^b$ and $U^* = (U^T U)^{-1} U^T$

HTML <canvas>

The HTML5 `<canvas>` element is a drawable bitmap region, which can be used to draw and script graphics using JavaScript. Its built-in `transform()` method only supports scale, translation and rotation. It also provides a native integration with the HTML `<video>` tag.

With canvas, we can create a JavaScript routine to manipulate a backprojected slide image *independently* from the video frame. This lets us overlay slide images over video frames directly in the client's browser.

System implementation

Define: τ as the threshold of the average re-projection error (in pixels).

For each frame:

1. Compute the coordinates of the slide corners in the frame, $\mathbf{p}$.
2. Estimate the affine transformation ($\boldsymbol{T}$) and the homography ($\boldsymbol{H}$) between the successive frames.
3. Backproject the corners using $\boldsymbol{T}$ to get the approximated coordinates, $\mathbf{q}$.
4. *If* $(\mathbf{q} - \mathbf{p}) < \tau$
   Then, use an affine transformation $\boldsymbol{T}$ for that frame.
   Else, use that frame's homography $\boldsymbol{H}$.

True backprojection vs. homography approximation

If the client has sufficient CPU resources	If the client cannot run computationally-intensive operations
Use: the direct homography computation	**Use:** the affine approximation
Bandwidth requirements: modified video, the slide images, timing and homography data.	**Bandwidth requirements:** modified video, the transformed slide images, timing data.

ID	# of slides	Slide deck size	# 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

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

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

The number of backprojected frames to send to the client based on the distance threshold τ in a test video. This test video had 117 slide transitions and a total of 71387 frames.

Video ID	WC			FB			LA		
Compression Setting (kbps)	600	400	200	600	400	200	600	400	200
Backprojected Video (Mb)	171.9	129.1	52.7	227.8	159.4	90.4	227.0	165.2	95.2
Modified Video (Mb)	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%

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.

Additional Information

Visit http://slic.arizona.edu to learn more about the SLIC project and to watch the demo.