Generation of lecture notes as images from recorded whiteboard and blackboard based presentations

Ondrej Jaribka*
Marek Šuppa†
Supervised by: Zuzana Černeková‡

Faculty of Mathematics, Physics and Informatics
Comenius University
Bratislava / Slovakia

Abstract

With raising amount of e-learning materials such as lecture videos or on-line video courses, we decided to develop an application, which can help students or content creators in their effort to prepare study materials. Main goal of our application is to create slides from given video depicting a black or white board without any occluding objects such as lecturer standing in front of this board. Slides will contain valid information from key frames of the given lecture video. Based on the assumption that the board is static in the video, this is done by extracting the board from video frames, which is then segmented into equally sized rectangular cells. These cells are stored and the change of information inside them is tracked. Afterwards, the final image is created from saved cells when all cells are sufficiently stable.

Keywords: whiteboard, blackboards, lecture, generation of images, slides, e-learning, video presentation, perceptual hashes, board extraction, key frame generation, slide generation

1 Introduction

With increasing amount of lectures, courses and other e-learning materials available on-line, it is becoming more and more apparent that students as the primary consumers of such content lack tools necessary for its usage in an effective fashion. Given the rise of massive open on-line courses [9] a strong push for creation of instructional videos can also be seen in academic environments. As stated in [12]: "fast expansion of the Internet and related technological advancements, in conjunction with limited budgets and social demands for improved access to higher education, has produced a substantial incentive for universities to introduce e-learning courses". In [11] the authors also claim that "many users stop their on-line learning after their initial experience". They further state that "instructor attitude toward e-Learning, e-Learning course flexibility, e-Learning course quality, *perceived usefulness*, *perceived ease of use*, and diversity in assessments are the critical factors affecting learners' perceived satisfaction".

It is not difficult to find inefficiencies in the ways instructional video content is most often consumed. For example it is very impractical to always rewind on-line lecture to get to the exact point where specific detail was discussed, pause the video, essentially "copy" the information from the paused video frame and then continue watching. Not only does it break the user's focus but it also requires additional interaction with the video content, that might result in undesired increase of the user's frustration, which is certainly not desired.

For this reason we decided to develop a tool that could help potential viewers extract information from these presentations or lectures into a more practical format. This could also help content creators to prepare and create content which can be then perceived more easily. As stated in [11] *technology* is one of the main factors affecting user satisfaction.

In our work, we focus on whiteboard and blackboard based lectures or presentations. The methods we present select key frames, from which a slide containing the information previously shown on board is created. Another requirement we enforce is that these slides should contain only information that is on the board, and therefore any occluding objects ought to be removed from the scene and information behind them should also be shown on our final slide.

This paper is structured as follows: in the *Related work* section we present an overview of academic literature, that touches the problems our work is designed to solve. Then we proceed to describe methods for *Board extraction*, *Removal of occluding objects* and *Change detection*. In the end, we present the results obtained thanks to our implementation of aforementioned methods in the *Results* section and finally conclude with a short summary of the most important results and possible directions for future work in

*o.jariabka@gmail.com
†marek@suppa.sk
‡zuzana.cernekova@fmph.uniba.sk

the *Conclusions and future work* section.

2 Related work

While our work tries to solve a very specific problem, there are a number of similar projects inspired by the increasing amount of readily available video-based lecture material.

Visual Transcripts [10] aims to create lecture notes from blackboard-style videos. Unlike our work, their system assumes that the video only has a blackboard in it, and that the video is static, except for content which is continually added and a mouse pointer which is used to highlight certain parts of the blackboard. The same type of video is used in another related work [8], in which the authors summarize the input video in form of a single image. Parts of the image function as links to positions in the input video and make it very easy to jump to the precise moment where a specific concept was first introduced.

The focus of many authors is mainly on one or few methods for specific subtasks, that represent the respective parts of our system. For example most papers on removing occlusion events from videos focus on 3D objects or use multiple cameras to generate the final image, as described in [4] and [3]. These methods are quite efficient, especially when implemented on GPUs. They might also be called accurate but they are not quite suited for our purpose.

On the other hand, many approaches are focused on creating fast and efficient methods for recognition of change in images and search for similar or unique objects in images. Various methods were designed in order to solve these tasks, such as using image features to detect similar objects in images or using perceptual hashes [13] to detect significant change. In [2], the authors developed a new method for image hashing, which can be used for fast look-ups of similar images. The thesis of Christoph Zauner [14] focuses on implementation and benchmarking of various image hashing algorithms and methods. While the primary aim of these methods was different, they can be easily adapted for the purpose of searching for change in information in series of frames generated from a video sequence.

Algorithms for detecting a change in scenes of video sequences were also proposed in [6] and [7]. These algorithms search for “drastical” points of change, such as hard cuts or special editing effects [1]. Even though these methods were quite useful as a model, we could not adapt them because our work is focused more on fine grained change between multiple frames and longer lasting sequences, where change is being slowly added through the span of multiple sets of frames.

[1]An example of thee effects are dissolve or wipe transitions.

3 Board extraction

Creators of instructional videos strive to give their content (be it on black or white board) the majority of visual space the video provides. Despite their efforts, often there are parts of video frames one would not expect to find in a “presentation slide”. Moreover, variability in these parts of video frames might cause issues in further stages of the processing pipeline, as it might be mistaken for the actual content. It follows, that in order to create a “presentation slide” from a set of video frames, only the significant parts of the frame need to be considered.

In this section, we describe main methods used for extracting board from video sequences. In our research we focused mainly on videos with one board present in the video or multiple boards not separated by a bigger gap in between them (Figure 1a). Before we detect the main region of interest where the board is located, we have to decide if we are looking at a white or black board. A simple preprocessing method ran on every input frame is used. This method first converts RGB image into grayscale and then computes a histogram. Since we focus only on white or black boards, the dominant colour from these is selected under the assumption that the area of a board occupies wast majority of given frame. This colour is then accentuated by thresholding the image to only extract colours that are close to our chosen colour. Thresholding value is different for every channel and it is approximated to previously found dominant color of our image. The mask obtained by this method is then used to compute bitwise `AND` in separated colour channels in order to boost our dominant colour (Figure 1b). After this process one of the proposed method is used to obtain main region of a board.

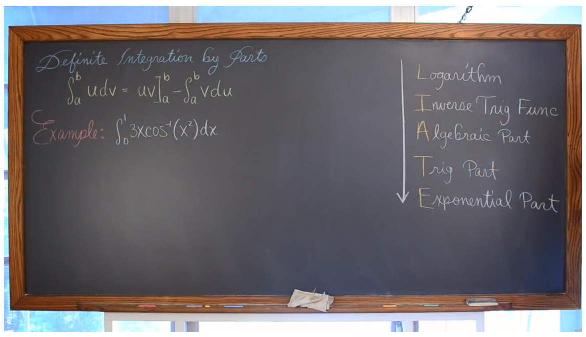
3.1 Histogram method

This method at first reduces colour spectrum of the image by thresholding the dominant colour one more time. In the next step, all the values in every row and column are summed up separately into two arrays. The first derivation is then computed on the acquired arrays to identify spikes in colour change. Extremes of these arrays then specify corners of the board. Finally, a bounding box around board is formed from the obtained points.

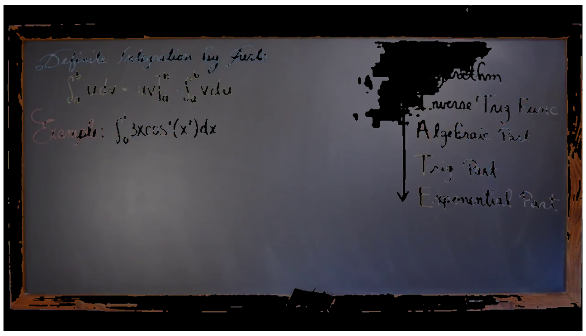
3.2 Region growing method

This method is based on a simple region growing algorithm. At first we have to select a seed for our algorithm. This is done by thresholding the main colour with almost maximum value found from the histogram computed in advance, as part of the preprocessing step. This highlights “blobs” of colour with highest values, so that we can be

[2]We used the video titled "Definite Integration by Parts" by Rob Tarrou which can be found at `https://www.youtube.com/watch?v=6rWG5WPysgE`



(a) Original image frame from the video.



(b) Output of the preprocesing function applied on the original frame. Note that green colour is accentuated while white or highly illuminated sections are attenuated.

Figure 1: Images of original video[2] frame and the resulting frame from the preprocesing function

almost sure that we are starting somewhere in the region of a board and not for example on the person standing in front of it. Then, we randomly pick one region where we place our seed. A simple region growing follows, in which we compare 4 neighbouring pixels to check if they fit our colour threshold (if they are part of the board). Once this is done, we obtain mask of a board. We search for contours and extract the biggest one. This contour is then returned as the bounding box of our board.

3.3 Processing of output from board extraction models

Output of every model is then submitted to the last test which checks if area occupied by the the bounding box is at least one third of a given image and if its shape is rectangular. Finally, the image is cropped. If a video or an image is taken from a slight angle, perspective of cropped board is then slightly shifted to compensate for this, so that the resulting slide would look more like an observer is standing in front of a board.

4 Removal of occluding objects

This section describes algorithms that we design in order to remove occluding objects from lecture videos. We consider as occluding object anything that is bigger than at least one third of a board and performs some sort of a move. An example of such an object can be a student or a lecturer. The main idea is to segment the board into smaller sections called cells and then keep track of how information changes in these cells by keeping a simple count of on how many frames we saw individual cell. This process can be described as a sequence of the following steps:

- Divide board into smaller cells
- Initialize each cell
- Compute mask of occluding objects
- Check each cell to see whether it is occluded by another object
- Change “seen” counter for each cell based on occlusion events
- Stitch individual cell into final slide

4.1 Initialization

We divide our cropped image of the extracted board into equally sized rectangular cells based on parameters of the input image. Individual cells are overlapping by values spanning between 10 to 20 pixels based on the size of a given frame. Each cell is then initialized by setting “seen” counter to 0 and its value is stored in an array. When cells are initialized, then we evaluate each of them to check if it is partly or fully occluded by another object[3]. This is done by computing bitwise AND between section and occlusion mask (see Section 4.2 and Section 4.3). If there is overlap detected the counter is not increased, otherwise it is incremented. For obtaining occlusion mask, we proposed two methods (Figure 2a).

4.2 Region growing based method

The First method is the same as our region growing algorithms since this method only considers pixels that fit our board condition. Pixels representing skin, person or other objects are omitted. This can then specify any objects that are in front of the board. If region growing algorithm was previously used for board extraction, mask produced by this method is then considered as mask of the board. This mask is then inverted and series of dilations and erosions is performed to remove error areas and to accentuate edges. Finally, we search for contours and filter out those, that are smaller than one third of the image.

[3]Note that if a cell is occluded throughout the whole video we believe that it is save to assume that there is no valid information behind it. Our software provides a way to leave out or add such a cell based on user input.

4.3 Absolute difference based method

The second one uses the last slide which was saved and therefore we assume, that this frame contains valid information and it is without any occlusion events. First, we calculate the absolute difference between current frame and this frame. This generates mask of events that changed from slide to slide. Then, we threshold this mask to remove pixels that arose from slight light changes for example person casting a shadow onto a background. Finally, similarly to the previous method we search for contours and filter out those, that are not at least one third of given image.

4.4 Final slide generation

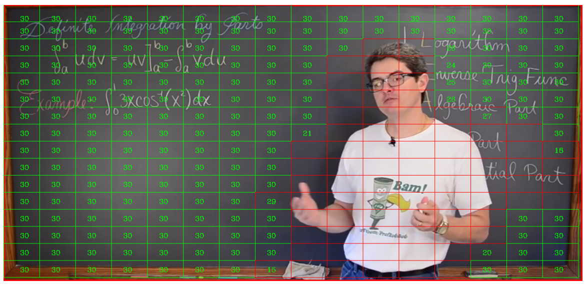
The final slide, shown in Figure 2b, is then "stitched" together from the last saved slide and currently stored sections where we threshold each section's counter with two values. If the counter value is higher then upper threshold we saw that section enough times to be sowed into the image. If the value is under lower threshold section is rejected and part of the old image is used. If the value is between these two thresholds one of the methods discussed in the section 5 is used to detected how much old and new section differ. If this similarity measure is higher then our threshold then section is rejected because sections where too similar and old part of the slide is used. If it is lower, the section changed enough so it can be sowed into the image.

5 Change detection

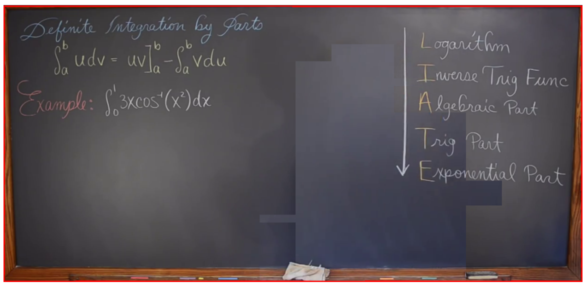
This section focuses on how are we actually going to choose key frames in the lecture or how we detect how much information in processed slides differ. The Main problem, that we faced was how we can choose key frame in the video or presentation. When we can say that enough information was added or subtracted from a board, so we can create slide of given board. For this we developed two main approaches: first one uses feature detection and second computes perceptual hash with one of the specified methods.

5.1 Feature detection

To detect change with this method we used ORB detector for feature detection. First step is to detect features in both last saved slide and current slide. Next, brute force matching, between features of these two image, is performed. Number of matched features is then compared to maximum of found matches. This value is then returned to be latter compared as our similarity measure for further tests.



(a) Input image segmented into grid of rectangular cells. Note that overlap of the individual cells is not shown on the image.



(b) Output of "stitching" algorithm without any further post-processing

Figure 2: This images show main steps in objects removal processing pipeline process.

5.2 Perceptual hashes

Another approach to detect change in our slides is through perceptual hashes. We implemented and compared probably three most known functions.

Average hash is a hashing function which computes hash of a file by firstly converting given image to grayscale. Then reducing a size of image to small square usually of size 8x8 to remove high frequencies. Hash is then created by computing mean value of pixels in transformed image which is then plugged into the thresholding function 1.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is} > \text{mean} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Difference hash or dHash, similarly to average hash, computes its value by firstly reducing a size of given image and converting into grayscale, then it calculates difference between adjacent pixels. This identifies relative gradient direction in the image. After this, it determines resulting value by using thresholding function.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is brighter than f(x-1,y)} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

While the first two algorithms are quick and easy they might be too rigid in comparison. For instance, it can generate false-misses if there is gamma correction or colour histogram applied to a image. To reduce this effect we use perceptual hash.

Main idea behind perceptual hash, or more commonly know as pHash, is that it uses discrete cosine transform (DCT) to reduce high frequencies in the image. Same as previous hashes initial step is to reduce size of given image and convert it into grayscale. Then it computes DCT on given image and subsequently reduces it to keep just lower frequencies of the picture. Next step is to calculate the mean DCT value while excluding the first term. This leaves out completely flat image information from being included. Finally, it further reduces DCT and computes resulting hash values based on the thresholding function similar to average hash.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is} > \text{mean DCT} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

These hash functions are used to compute hashes of the last saved slide and the currently processed slide. Hamming distance is then computed between these hashes. To unify the output from these functions and feature detection function we compare length of last hash and hamming distance.

Output from these similarity functions is then compared to our similarity threshold. If our measure is lower than this threshold, slides differ and information on the board have to be different, so slide is created. If it is greater, then slides are similar. Current slide is thrown away and all counters are reset.

This section summarizes the results of the proposed methods. We created a dataset of videos and presentations from various on-line lectures and courses. We chose to implement our methods in Python programming language using an Open source computer vision library - OpenCV [1]. The tests run on single desktop workstation, using the following hardware: NVIDIA Corporation GeForce GT 650M, Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz. Preprocesing of every frame in a video, took on average with 300 method calls - 7.02 ms (+- 0.31 ms). Every method was tested for performance from 300 method calls by measuring its individual speed in milliseconds in our processing pipeline.

5.3 Board extraction results

In this section, we present performance values as well as precision and recall values for individual methods.

Method name	speed in ms	sdv
Histogram	4.75	0.65
Region growing	**2.03**	0.57

Table 1: Average performance values in milliseconds with standard deviation for individual methods

As we can see in Table 1, the Histogram algorithm was on average about two times slower than the region growing algorithm. This is understandable as the histogram algorithm is much more complex and needs to perform more operations than a simple region growing algorithm.

While this comparison might be interesting it only shows how fast the respective methods are. In order to evaluate the system as a whole we are more interested in their performance: essentially the answer to the question how well were these two methods able to extract the board from input images.

In order to answer that question, we used a dataset of 19110 images. For these images the bounding box of the board was marked by a human expert. This should serve as the “ground truth” in our experiments [5].

To compare these two methods we compute *precision* and *recall* for both of them. We define *precision* as:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{4}$$

and recall as:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{5}$$

In both of these equations *true positives* are defined as the area of the image which was marked by the method as a board and the “ground truth” agrees with that. *False positives* is defined as the area that was marked by the method as a board but the “ground truth” did not mark it as a board and *false negatives* is the area which was marked by “ground truth” as a board but the method does not agree with that.

These two values can be put together into a single metric that is called *F1 score* that is defined as the harmonic mean of *precision* and *recall*:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6}$$

Note that while these methods are traditionally more often used in information extraction, they are also considered a well defined metric in computer vision and image processing [5].

Method name	precision	recall	F1 score
Histogram	0.4245	**0.9828**	0.5929
Reg. growing	**0.9899**	0.9356	**0.9620**

Table 2: Precision, recall and F1 score values of board extraction methods.

Given these metrics, we can observe some interesting statistics about the proposed methods in Table 2. We can see that while the histogram algorithm has a very high recall and therefore produced quite few false negatives its precision is quite low on the other hand. This suggests that it produced quite a lot of false positives which might not be desired for the final processing pipeline.

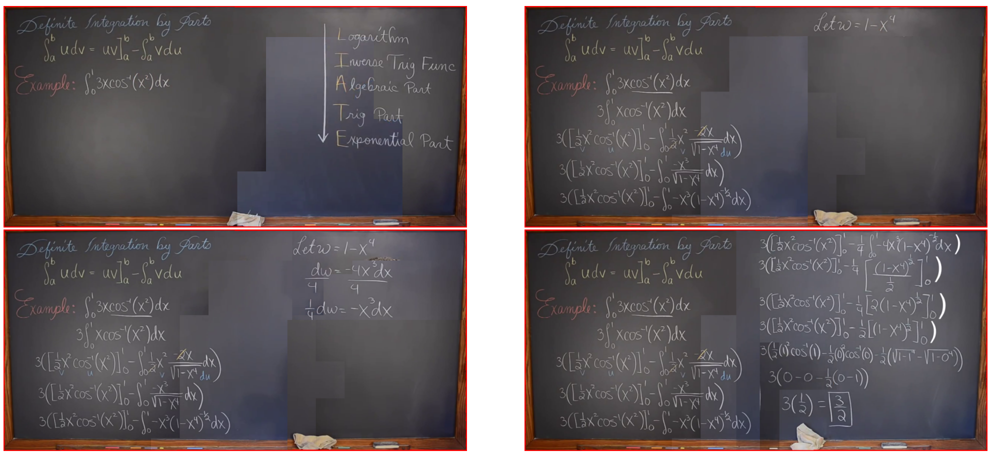


Figure 3: Output images obtained by running the final processing pipeline on an example video.

The region growing algorithm on the other hand shows balanced values for both precision and recall. This suggests that it is a robust method, even though it produced an increased amount of false negatives which might not be desirable, since it would mean that parts of the board would be lost.

As we can see from the comparison of F1 scores, the region growing algorithm seems to be a more robust method and therefore we can conclude that it might be a better choice than the histogram based method.

5.4 Object removal results

While in the previous section the metric for performance comparison of methods was quite straightforward, finding one for object removal has proven difficult. In the end we used a hand-curated dataset of 150 images for which the ground truth (object to be removed) was provided by a human expert.

Method name	precision	recall	F1 score
Reg. growing-based	0.2531	0.5595	0.3485
Abs diff-based	**0.4767**	**0.9877**	**0.6430**

Table 3: Precision, recall and F1 score values of board extraction methods.

The precision, recall and F1 score value for both of the suggested methods can be found in table 3. As we can see, the scores of the absolute difference-based method are better. This can be explained by the nature of the region growing-based method: given its randomized seeding, it might easily start growing the board's region from a point in the middle of the object, that should be removed. This essentially means, that it does the inverse task and produces incorrect result. Looking at the performance values in table 4 we can see that while region growing-based method is faster, the difference is not substantial. Based on this analysis we chose the absolute difference-based method to be used in the final pipeline.

Name	speed in ms	sdv
Reg. growing-based	**160.47**	0.91
Abs diff-based	195.68	0.84

Table 4: Average performance values in milliseconds with standard deviation for individual methods

5.5 Change detection results

It was difficult to design a measure that would express how well is function performing in terms of selecting key frames in video. This was due to the fact, that even though we can have individual frames in sequence tagged the resulting value, if the selected key frame is in the right place, is very perception dependent.This is why we chose to only measure how many frames will a method create and how long will it take the whole pipeline (with a given measure) to do so.

Name	speed in ms	sdv
ORB algorithm	13.13	0.85
pHash algorithm	7.52	0.76
dHash algorithm	5.32	0.43
aHash algorithm	**3.23**	0.59

Table 5: Average performance values in milliseconds with standard deviation for individual methods

As we can see in Table 5, the ORB algorithm was the slowest which is understandable as it is also the most complex algorithm. Performance of hashing methods was comparable with fastest one being the aHash algorithm[4] This can be also closely related to values in Table 6 where, aHash not only has the best performance values but it also managed to create most slides. This can be associated with aHash being one of the the simplest methods that is very vulnerable to even slight light intensity change in the

[4]Which is not very surprising since it performs simple mean on given images.

scene. In a similar fashion, dHash suffers from the same problem, as it also created the same number of slides.

Name	speed (sec)	# of slides
ORB algorithm	1783	7
pHash algorithm	1117	**6**
dHash algorithm	1115	8
aHash algorithm	**1112**	8

Table 6: Average performance values in seconds for individual runs of entire pipeline with given method on the whole input video and the number of slides created using these methods.

From values in tables 5 and 6 we can conclude that pHash is the best choice for detecting change in our video sequences. It provides the best ratio between speed of individual methods, speed of the entire run of a pipeline and number of created slides. While feature detection did reasonably well in comparing images, its performance could not compare to hashing algorithms.

6 Conclusion and future work

In our work, we present and compare methods for detecting and extracting boards from white or black board video based presentation. This setup can later be abstracted to detecting and extracting large scale object in image or video that matches some sort of a predicate. In our case, it was colour and shape of an object. We got sufficient results with regards to board extraction with the best method being region growing.

We also tested and evaluated functions for comparing how similar are two images in order to generate key frames in the video sequence. Finally, we proposed methods for extracting information from given presentation even with occluding events present in the sequence. In this category, we also managed to achieve sufficient results with hashing functions with the best one being the pHash algorithm.

We proposed two methods for removing objects in front of our board. The performance of the absolute difference-based one was found to be better overall and chosen for the final pipeline. The slides created by our final processing pipeline can be seen in 3.

Many of these algorithms can be improved. For example after function that creates final slide, we can use post processing method to smooth transition between sections of the old image and the new one. Or it might be possible to further shift perspective in order to better fit the current slide if for instance camera moved during the presentation. Support for multiple boards with bigger gaps between them can be added. Also based on colour of a board, text on slide can be further enhanced for latter used in OCR algorithm.

Given our focus on simplicity, performance and speed, we also believe that the proposed algorithms might serve as a basis for a system, that would produce slides as images from white and black board based videos in real time.

References

[1] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[2] Cédric De Roover, Christophe De Vleeschouwer, Frédéric Lefèbvre, and Benoit Macq. Robust image hashing based on radial variance of pixels. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–77. IEEE, 2005.

[3] Takahide Hosokawa, Songkran Jarusirisawad, and Hideo Saito. Online video synthesis for removing occluding objects using multiple uncalibrated cameras via plane sweep algorithm. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–8. IEEE, 2009.

[4] Byung-Gook Lee, Ho-Hyun Kang, and Eun-Soo Kim. Occlusion removal method of partially occluded object using variance in computational integral imaging. *3D Research*, 1(2):6–10, 2010.

[5] Vladimir Y Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David Doermann, and Thomas Drayer. Performance evaluation of object detection algorithms. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 965–969. IEEE, 2002.

[6] Jianhao Meng, Yujen Juan, and Shih-Fu Chang. Scene change detection in an mpeg-compressed video sequence. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 14–25. International Society for Optics and Photonics, 1995.

[7] Takafumi Miyatake, Satoshi Yoshizawa, and Hirotada Ueda. Method for detecting change points in motion picture images, January 28 1992. US Patent 5,083,860.

[8] Toni-Jan Keith Palma Monserrat, Shengdong Zhao, Kevin McGee, and Anshul Vikram Pandey. Notevideo: facilitating navigation of blackboard-style lecture videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1139–1148. ACM, 2013.

[9] Laura Pappano. The year of the mooc. *The New York Times*, 2(12):2012, 2012.

[10] Hijung Valentina Shin, Floraine Berthouzoz, Wilmot Li, and Frédo Durand. Visual transcripts: lecture notes from blackboard-style lecture videos. *ACM Transactions on Graphics (TOG)*, 34(6):240, 2015.

[11] Pei-Chen Sun, Ray J Tsai, Glenn Finger, Yueh-Yang Chen, and Dowming Yeh. What drives a successful e-learning? an empirical investigation of the critical factors influencing learner satisfaction. *Computers & education*, 50(4):1183–1202, 2008.

[12] Thierry Volery and Deborah Lord. Critical success factors in online education. *International Journal of Educational Management*, 14(5):216–223, 2000.

[13] Tom Yeh, Konrad Tollmar, and Trevor Darrell. Searching the web with mobile images for location recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–76. IEEE, 2004.

[14] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. Master's thesis, FH Hagenberg, 2010.