UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ROSÁLIA GALIAZZI SCHNEIDER

Panoramic e-Learning Videos for non-Linear Navigation

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Manuel Menezes de Oliveira Neto
Advisor

Porto Alegre, May 2013

CIP – CATALOGING-IN-PUBLICATION

Galiazzi Schneider, Rosália

Panoramic e-Learning Videos for non-Linear Navigation / Rosália Galiazzi Schneider. – Porto Alegre: PPGC da UFRGS, 2013.

84 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Manuel Menezes de Oliveira Neto.

1. Image registration. 2. E-learning videos. 3. Computer vision. I. Oliveira Neto, Manuel Menezes de. II. Title.

ACKNOWLEDGEMENTS

I would like to thank specially my advisor, Manuel Menezes de Oliveira Neto. More than anything else, he taught me to work hard and to strive for perfection. We disagree in a variety of things, but we do not disagree in our love of science and in our belief that we can be a part of change. We will keep fighting the same war, now in different front lines for a while. I would also like to thank Mark Pauly, who gave me the opportunity to be an intern at his lab, and Sofien Bouaziz who taught me how to be the master of my own research.

I thank Renan Sales Barros, André Luís Beling da Rosa, Vinícius Azevedo, Leticia Tonetto, Carina Loureiro Andrade, Tatiana Cartagena, Frederico Limberger, Bernardo Henz, Daniele Fernandes, Vitor Augusto Jorge, Lucyana Vulcão, Guilherme Oliveira, Jerônimo G. Grandi, Tales Miranda, Hernandi Filho, Victor Adriel Oliveira, Victor Chitolina Schetinger, and the whole Computer Graphics Group for sharing with me this first step of the difficult journey to become a researcher. We have an amazing group and I hope to see you all again in the promising future that stands before us.

Also, I could not forget to thank my friends Daniela Pereira, Bruna Dias, Fernanda Mattos, Johana Pires, Lara Torrada Pereira, Fátima Balenti and Júlia Cousin, for supporting me not only in these two years but in the last fifteen.

Finally, I would like to thank most of all my family for their unconditional love and support. My parents, Maria do Carmo Galiazzi and André Martin Schneider, my extra pair of parents, Rosane Machado de Azevedo and Manuel Haimovici and my siblings Luíza Galiazzi Schneider and José Luciano Galiazzi Haimovici. Also, I would like to thank the support of those who were part of my family for most of these two years, Fernando de Avila Bottin, Maria da Graça, Fernando G., Samara, Samanta and Sabrina Bottin.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This thesis introduces a new interface for augmenting existing e-learning videos with panoramic frames and content-based non-linear navigation. In conventional e-learning videos, each frame is constrained to the subset of the lecture content captured by the camera or frame grabber at that moment. This makes it harder for users to quickly revisit and check previously shown subjects, which might be crucial for understanding subsequent concepts. Locating previously seen materials in pre-recorded videos requires one to perform visual inspection by sequentially navigating through time, which can be distracting and time-consuming.

We augment e-learning videos to provide users direct access to all previously shown content through a simple pointing interface. This is achieved by automatically detecting relevant features in the videos as they play, and assigning them hyperlinks to a buffered version in a completely transparent way. The interface gradually builds panoramic video frames displaying all previously shown content. The user can then navigate through the video in a non-linear way by directly clicking over the content, as opposed to using a conventional time slider. As an additional feature, the final panorama can be exported as a set of annotated lecture notes.

We demonstrate the effectiveness of our approach by successfully applying it to three representative styles of e-learning videos: Khan Academy, Coursera, and conventional lecture recorded with a camera. We show that we can achieve real-time performance for low-resolution videos (e.g., 320x240) on a single desktop PC. For higher resolution videos, some pre-processing is required for feature detection (using SIFT). However, since the most expensive parts of our processing pipeline are highly parallel, we believe that real-time performance might be soon achievable even for full HD resolution.

The techniques described in this thesis provide more efficient ways for exploring the benefits of e-learning videos. As such, they have the potential to impact education by providing more customizable learning experiences for millions of e-learners around the world.

Keywords: Image registration, e-learning videos, computer vision.

RESUMO

Este trabalho introduz uma interface para estender vídeos educacionais com panoramas e navegação não-linear baseada em conteúdo. Em vídeos de e-learning convencionais, cada quadro está restrito ao subconjunto da cena capturado naquele momento. Isso torna difícil para o usuário revisitar conteúdos mostrados anteriormente, que podem ser essenciais para o entendimento dos conceitos seguintes. Localizar conteúdos anteriores nesses vídeos requer uma navegação linear no tempo, o que pode ser ineficiente.

Estendemos vídeo-aulas para prover ao usuário o acesso direto a todo o conteúdo apresentado através de uma simples interface. Isso é feito pela detecção automática de pontos relevantes no vídeo e a criação de hyperlinks a partir desses pontos de maneira completamente transparente. Nossa interface constrói gradualmente um panorama clicável que mostra todo o conteúdo visto no vídeo até o dado momento. O usuário pode navegar pelo vídeo simplesmente clicando no conteúdo desejado, ao invés de utilizar a tradicional barra deslizante de tempo. Nosso panorama também pode ser exportado no final da execução, juntamente com anotações feitas pelo usuário, como um conjunto de notas de aula.

A eficiência da nossa técnica foi demonstrada com a aplicação bem-sucedida a três categorias de vídeos que são representativas de todo o conjunto de vídeo-aulas disponíveis: Khan Academy, Coursera e aulas convencionais gravadas com uma câmera. Demonstramos que foi possível atingir os resultados em tempo real para vídeos de baixa resolução (320x240). No caso de resoluções mais altas, é necessário que a detecção de features (usando SIFT) seja feita em uma fase de pré-processamento. Como a parte mais custosa do nosso pipeline é extremamente paralelizável, acreditamos que a execução de vídeos de alta resolução em tempo real seja um resultado alcançável em curto prazo.

As técnicas descritas nessa dissertação disponibilizam maneiras mais eficientes de explorar vídeos educacionais. Dessa forma, elas tem potencial para impactar a educação, disponibilizando experiências educacionais mais customizáveis para milhões de estudantes em todo o mundo.

Palavras-chave: Vídeos Educacionais Aumentados, Visão Computacional, Registro de Imagens.

1 INTRODUCTION

There is a recent trend towards the online publication of educational material from universities around the world. Websites like Coursera (COURSERA, 2013), edX (EDX, 2013), MIT OCW (MIT OCW, 2013) and Khan Academy (KHAN ACADEMY, 2013) are only some examples of a huge movement to make world-class education available to everyone, for free. The impact is huge: top universities are providing high-quality educational content to millions of students worldwide.

A recent study conducted on the student perception of instructor-made videos in online education indicates extremely positive evaluations of this type of strategy (ROSE, 2009). The most famous online education websites all rely deeply on videos. Examples of different types of such videos can be seen in Figure 1.1. Videos have advantages even when compared to face-to-face lectures. Pausing, repeating, skipping - those are all features that make the watching experience more customizable and effective.

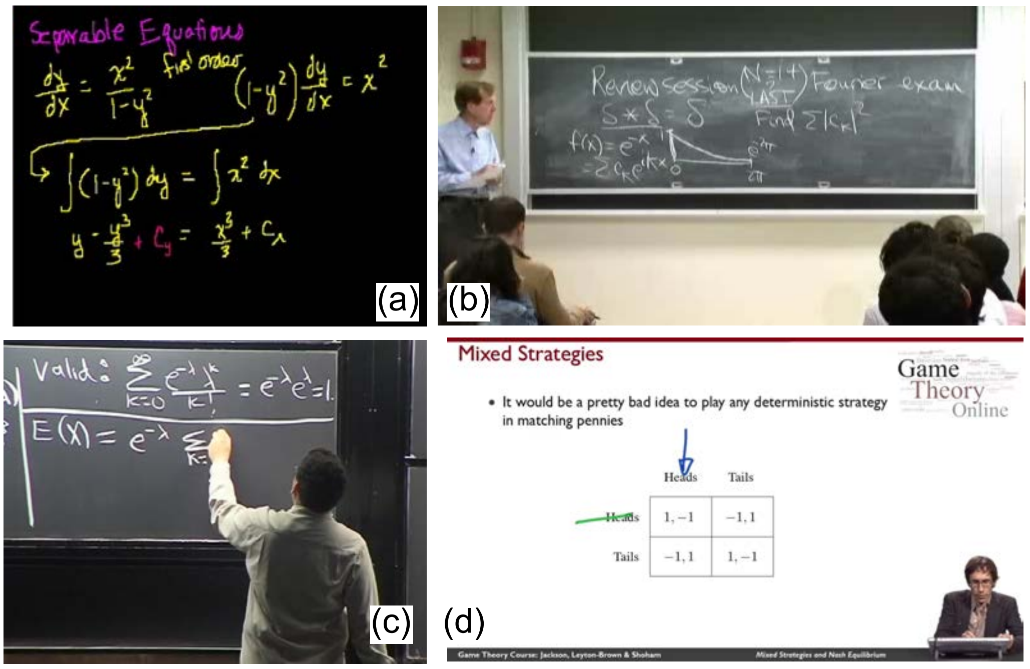

(a)
(b)
(c)
(d)
Separable Equations
Mixed Strategies
It would be a pretty bad idea to play any deterministic strategy in matching pennies
Game Theory Online

Figure 1.1: Videos are an essential part of online education. (a) Lecture on Separable Equations, Khan Academy (KHAN ACADEMY, 2013). (b) Video from Probability, on iTunesU (ITUNESU, 2013) (c) Video from Computational Science and Engineering I, MIT OCW (MIT OCW, 2013). (d) Video from Game Theory course, on Coursera (COURSERA, 2013).

Unfortunately, although one can pause and repeat sections of the video as he/she pleases, finding the points of interest requires sequential navigation with a time slider. While this is not critical when repeating concepts just presented, it tends to be far less convenient when one wants to revisit or double check a concept presented much earlier, but which is key for understanding the subject being explained at the moment. It is often hard to remember the point in time when the instructor presented a particular concept, or when an equation was introduced and explained. In those situations, navigating with a time slider tends to slow down the interaction and introduce distractions (as the user is forced to see different concepts along the way). A much natural way of indicating the subjects to be reviewed is to directly point over the equation or the word(s) representing the concept to be reviewed. However, this will require creating a visual record of all content presented along the lecture.

To achieve this goal, we create a panoramic representation of the video. Suppose the camera focuses on a given part of the board. There is no reason why the user should not be able to see the rest of the room, if this information can somehow be extracted. This panorama can also be used to generate notes of the lecture. Also, the panorama is a new dimension - we can use it, for example, to add notes to a given part of the video that is not related to the time. The availability of such a panorama also allows us to directly go the explanations of the displayed content. For this, one can use hyperlinks connecting the contents the user sees to the exact points in the video where they were introduced. To the best of our knowledge, our work is the first to propose this navigation style. Figure 1.2 illustrates our proposed extensions.

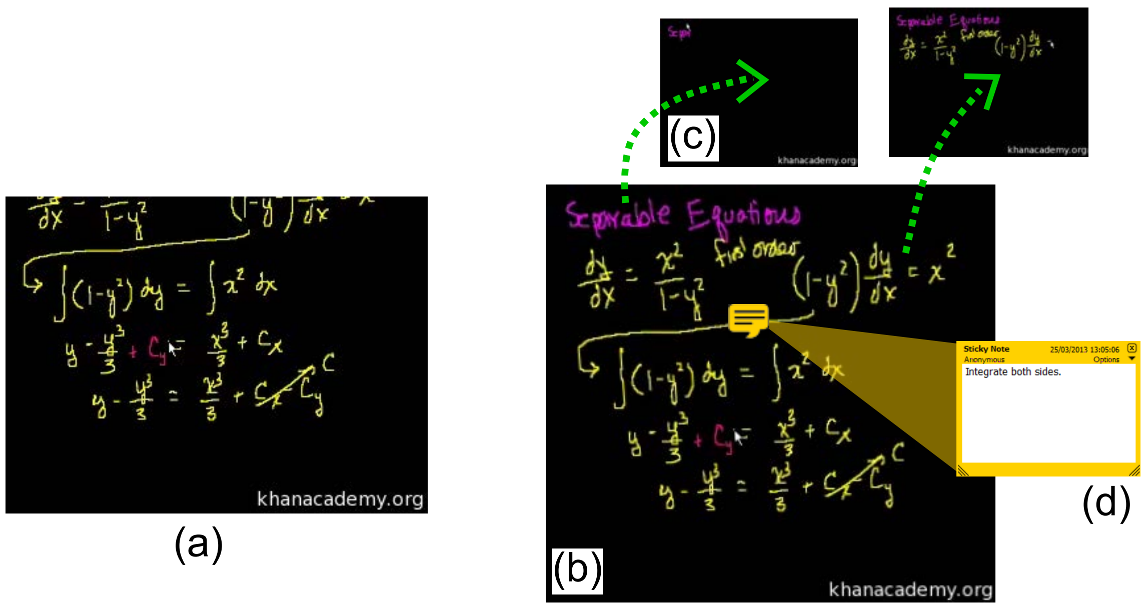


Figure 1.2: Interface first draft. (a) Regular frame at time 4:43. (b) Extended panoramic frame at time 4:43. (c) Hyperlinks allowing direct navigation to times 0:30 and 1:47. (d) Note added to video.

One of the main challenges is the variety of styles of e-learning videos available on the web. We want our technique to be powerful, yet general enough to be applicable to many videos. We explored three types of videos, that we believe cover the great majority of lectures currently available on the internet: Khan Academy's tablet videos (Figure 1.1(a)), Coursera's slide-based videos (Figure 1.1(d)) and regular video lectures (Figure 1.1(b)(c)). We will discuss the specifities of each type in Chapter 2. These three categories are very different from each other, and each imposes its own challenges to the creation of our interface. The applicability of our technique to these three categories does demonstrate that it not only covers the current scenario, but that it could probably be extended to new types of videos that may appear in the future.

1.1 Thesis Statement

It is possible to augment e-learning videos with panoramic frames and hyperlinks to allow content-based non-linear video navigation and exploration using a pointing interface. Such hyperlinks can be created on the fly, during the video watch/streaming, providing direct access to all viewed content up to that moment. As they are created, these hyperlinks provide an interface for exploring the video content in a non-linear way. Moreover, the videos can receive interactive annotations, which together with the hyperlinks, can be created and shared in a collaborative way by a group of people over the Internet. One can also automatically generate notes for the e-lectures.

To demonstrate this thesis, a few challenges must be overcome. First, we need to register video frames to allow the creation of a panorama. Second, we need to be able to find features that allow one to identify the instant in the video where each concept first appeared. Finally, we need to apply these operations in a broad range of existing videos.

I demonstrate it by creating a system that performs these operations in three very different categories of e-learning videos.

1.2 Structure of this Thesis

The remaining of this thesis is organized as follows: Chapter 2 discusses the different e-learning systems available nowadays. In Chapter 3, we review the existent research in the area of augmenting e-learning videos. We cover three main areas where we believe our work fits, in terms of the possibilities it creates for interaction. The next two chapters are dedicated to the background in Computer Vision needed to implement our system. In particular, Chapter 4 discusses the techniques for feature detection and matching used in this work, as well as the methods for image registration. Chapter 4.4 goes further into Computer Vision techniques to discuss the method used to track objects in the videos and the techniques for line detection used to automatically detect the limits of boards. In Chapter 5, we discuss our implementation of the proposed interface. Chapter 6 presents some results. Chapter 7 summarizes this thesis and discusses directions for future exploration.

2 ONLINE EDUCATION SYSTEMS

In the last few years, there has been a trend of top universities making educational resources available online, for free. Most recently, with the advent of Massive Open Online Courses - open online courses aimed at large-scale participation that have start and end dates - this process has accelerated. The biggest sites have millions of students from almost every country in the world. In this Chapter, we discuss the main existing paradigms for online education.

Just like normal courses, online courses come in very different shapes. Some include discussion forums, problem sets and final exams, while some are only a collection of videos. Since we are mostly interested in the video lectures, we will divide our discussion into three main categories, according to the kind of video provided: canvas-based videos, slide-based videos, and recorded lectures. We believe most existing e-learning systems can be mapped into one of these three categories.

The following sections provide a short review of the main e-learning systems that are part of these three categories. We will, however, restrict ourselves to briefly introducing the existing systems and discussing the type of video lectures they provide. A broader discussion of the impacts of MOOCs in higher education and the historical development of the current scenario are out of the scope of this thesis and can be found in a report issued recently by Powell and Yuan (POWELL; YUAN, 2013).

2.1 Canvas-based videos

We define a video as canvas-based when the content shown can be seen as part of a canvas where the information is drawn. The elements are typically text, equations, and drawing made with some sort of stylus. This notion of a canvas is essential - the spatial relation of the frames to each other can be easily inferred from it, which is very important for our technique. In these videos, the background is usually of a fixed color, and the information is drawn with high contrast relative to it. Also, because these are usually explanations of a single concept, they tend not to be very long. These videos are the most easily adaptable to our interface.

2.1.1 Khan Academy

Khan Academy, a not-for-profit organization created in 2006 by Salman Khan (KHAN ACADEMY, 2013), is the biggest representative of canvas-based videos. It provides a large number of short video lectures about a huge variety of subjects. This is different than most other e-learning methods, that provide whole courses. The videos usually start with a black canvas in which the text, equations, and drawings are progressively written with a stylus. They are usually no longer than 15 minutes. Examples can be seen in Figure 2.4.

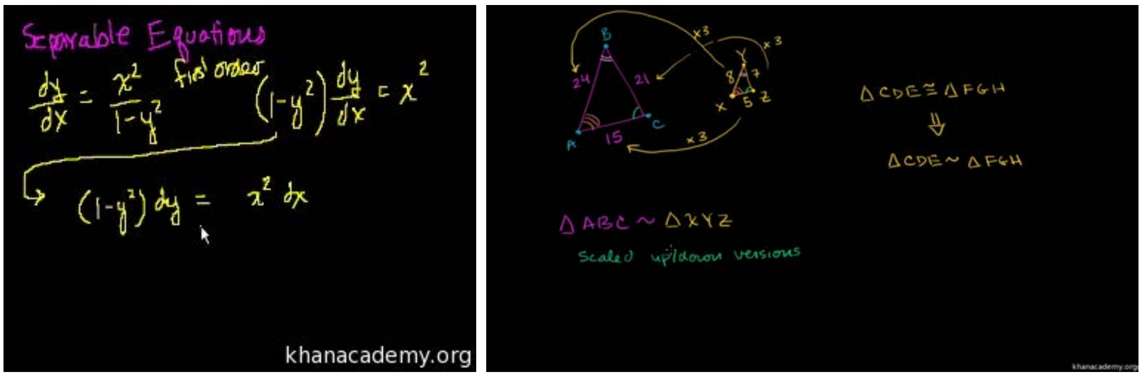
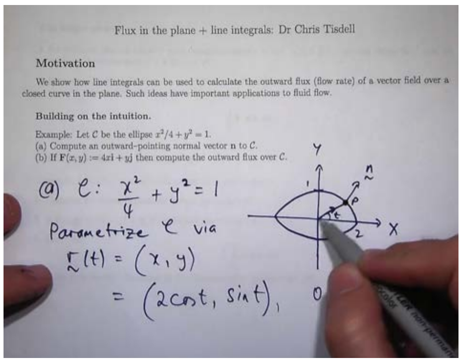

Separable Equations
first order
$\frac{dy}{dx} = \frac{x^2}{1-y^2}$
$(1-y^2)\frac{dy}{dx} = x^2$
$(1-y^2)\,dy = x^2\,dx$
khanacademy.org
$\triangle CDE \cong \triangle FGH$
$\triangle CDE \sim \triangle FGH$
$\triangle ABC \sim \triangle XYZ$
Scaled up/down versions
khanacademy.org

Figure 2.1: Examples of Khan Academy videos. (a) Differential equations lecture. (b) Video on the similarity of triangles showing drawings instead of equations.

2.1.2 Independent videos

Many e-learning videos that are not part of courses in a bigger system - use the same style as Khan Academy. A different approach that can also be seen as belonging to the canvas-based category are videos that show the solution of mathematical problems on paper. Tisdell's series "Understand Mathematics" (UNDERSTAND MATHEMATICS, 2013) (Figure 2.2)) is a good example of independent videos.



Flux in the plane + line integrals: Dr Chris Tisdell

Motivation

We show how line integrals can be used to calculate the outward flux (flow rate) of a vector field over a closed curve in the plane. Such ideas have important applications to fluid flow.

Building on the intuition.

Example: Let C be the ellipse $x^2/4 + y^2 = 1$.
(a) Compute an outward-pointing normal vector $\mathbf{n}$ to C.
(b) If $\mathbf{F}(x,y) := 4x\mathbf{i} + y\mathbf{j}$ then compute the outward flux over C.

(a) $\mathcal{C}: \frac{x^2}{4} + y^2 = 1$
Parametrize $\mathcal{C}$ via
$\underset{\sim}{r}(t) = (x, y)$
$= (2\cos t, \sin t),$
y
x
$\underset{\sim}{n}$
P
t

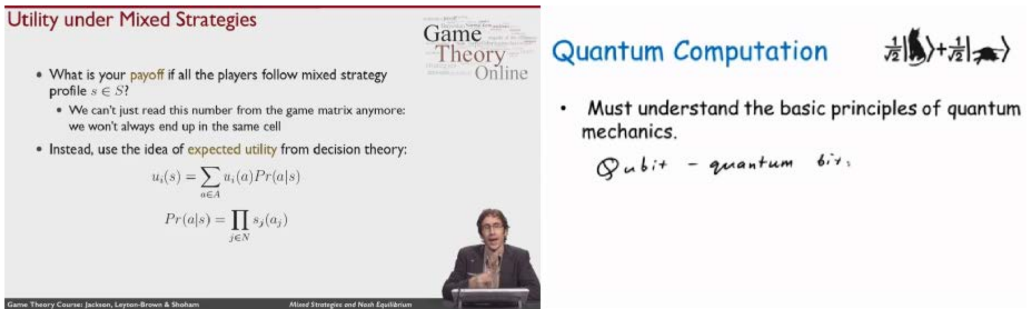
Figure 2.2: Problem solving videos can be seen as canvas-based. Chris Tisdell's "'Understand Mathematics"' (UNDERSTAND MATHEMATICS, 2013).

2.2 Slide-based videos

In the slide-based category, the background of the video is composed by a set of slides. On top of it, the instructor makes notes or marks specially important parts. Occasionally, the instructor is also shown in a small portion of the slide. The most important difference from these videos to the canvas-based ones is that there is no explicit spatial relation between the frames. While one can imagine the slides as being side-by-side, the information of the global position of each frame is not available. To allow our system to deal with these videos, we artificially give each slide a place in a global coordinate system. Since the videos are originally created from a set of slides, we can easily export these slides and create a global map putting them side-by-side.

2.2.1 Cousera

Coursera is the biggest online educational site available today. It was created by two professors from the Stanford University, Andrew Ng and Daphne Koller, in 2012. As of March 2013, over 300 courses from 62 partner universities were available, in five different languages. Coursera's videos are mostly slide-based, and the website also provides the slide notes, which we use for the registration task. Examples of two Coursera's slide-based videos can be seen in Figure 2.3(a).



Utility under Mixed Strategies

Game Theory Online

- What is your payoff if all the players follow mixed strategy profile $s \in S$?
  - We can't just read this number from the game matrix anymore: we won't always end up in the same cell
- Instead, use the idea of expected utility from decision theory:

$$u_i(s) = \sum_{a \in A} u_i(a) Pr(a|s)$$

$$Pr(a|s) = \prod_{j \in N} s_j(a_j)$$

Game Theory Course: Jackson, Leyton-Brown & Shoham

Mixed Strategies and Nash Equilibrium

Quantum Computation

- Must understand the basic principles of quantum mechanics.

Qubit - quantum bit,

Figure 2.3: Examples of slide-based videos. (a) Game Theory course on Coursera. (b) Quantum Computation course on edX

2.2.2 Udacity

Udacity is a for-profit company founded by Stanford University professors Sebastian Thrun, David Stavens and Mike Sokolsky. Udacity differs a little bit from other online course providers in that it does not partner with universities; instead, it offers a small amount of self-created courses. The type of video in Udacity also does not fit exactly in the concept of slide-based, they are not necessarily coming from a set of slides. Instead, the background may largely vary along the video.

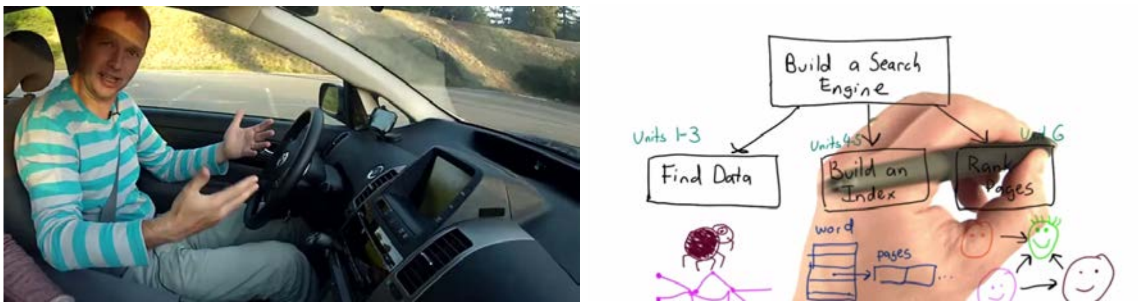


Figure 2.4: Examples of Udacity videos. (a) Artificial Intelligence lecture. (b) Introduction to Computer Science.

2.2.3 edX

EdX is an open source online platform, thus differing from other MOOC providers, which are commercial enterprises. It was founded by the Massachussets Institute of Technology and Harvard University. EdX also offers courses from the University of Berkeley. Other prestigious universities should join the initiative in 2014. In terms of video lectures, the courses provided are very similar to Coursera, as can be seen in Figure 2.3(b).

2.3 Recorded conventional lectures

One of the most common ways to make lectures available online is by simply uploading recorded lectures. While that is not the mostly adopted type of video for new courses, many universities have done that in the past. Thus, there is a huge amount of legacy video that follows this style. The variability is also huge, making the automatic handling of all such videos a very difficult problem. This type of video certainly poses the biggest challenge to our interface.

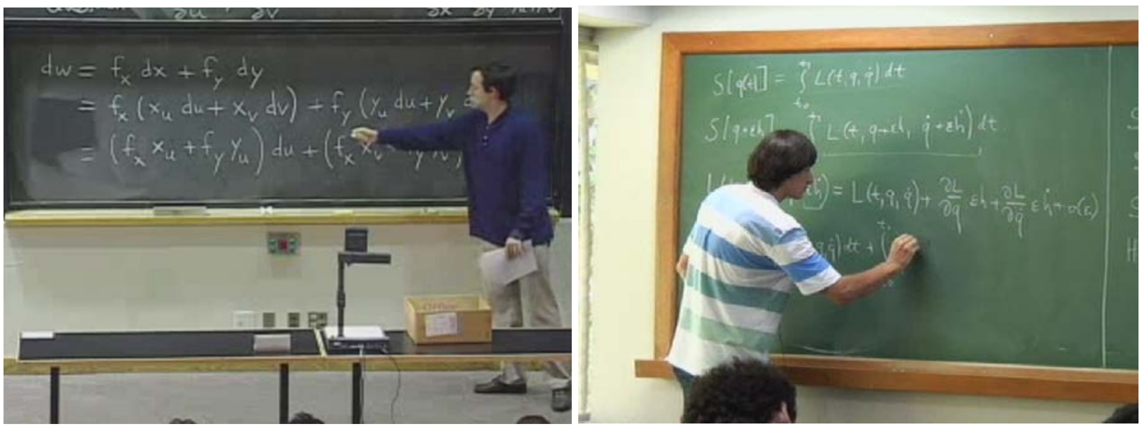


Figure 2.5: Examples of recorded video lectures. (a) MIT OpenCourseWare course on Multivariable Calculus. (b) Introduction to Wave Theory at the Institute of Pure and Applied Mathematics (IMPA).

The first problem is defining exactly which parts of the video have important content, and which parts should be ignored. To detect the boards in the image of the entire room, we have to rely on line detection algorithms. The state-of-art in line detection is often not robust enough, specially when dealing with low-resolution and noise.

A second problem is that, even considering a video where the boards are perfectly detected, the instructor or the students might occlude it at some points. To achieve the same results obtained with unoccluded styles, such as slide-based and canvas-based videos, one needs to be able to detect and remove unwanted (moving) objects, and fill in the resulting "holes". These are difficult tasks for which robust algorithms are still an open problem in Computer Vision.

Finally, registering frames to a global coordinate system is more complicated than for other types of videos. While Khan Academy uses a black background and text is written in bright colors, recorded lectures often do not have enough features to allow reliable mapping from one frame to the other. Also, video lengths are usually over 40min, which makes cumulative error an enormous issue. In addition, some boards can be moved, making registration literally impossible in some cases. If a board covers a big part of the video frame, registration cannot distinguish between board movement and camera movement in the opposite direction.

2.3.1 MIT OpenCourseWare

One of the first websites to have video content online was MIT OpenCourseWare (MIT OCW), which was launched in 2002. MIT OpenCourseWare is essentially an attempt from the Massachussets Institute of Technology to make the material from its undergraduate- and graduate-level courses freely available to everyone. Differently from other websites, OCW videos are not part of online courses, but materials from face-to-face courses made available online. As such, the video lectures provided are simply regular recorded lectures.

2.3.2 Legacy Lectures

While Udacity and Khan Academy offer their own content and Coursera partners with a selected group of universities, hundreds of other institutions around the world also move towards making education more accessible to everyone. Recording regular lectures is the simplest and cheapest way of making a course available. For this reason, this is probably still the most common type of e-learning video on the Internet. Many prestigious universities have resources available in this format. Also, some regionally recognized institutions have taken this path. In Brazil, one such example is the National Institute of Pure and Applied Mathematics (IMPA) (IMPA - INSTITUTO NACIONAL DE MATEMATICA PURA E APLICADA, 2013), that provides math lectures from the graduation courses. In India, many courses from technological institutes are available with the National Programme on Technology Enhanced Learning (NPTEL) (NATIONAL PROGRAMME ON TECHNOLOGY ENHANCED LEARNING, 2013).

2.4 Summary

This Chapter briefly reviewed the main current e-learning systems. It highlighted the differences between them, specially considering the type of videos they offer. It classified such systems according to three paradigms for generating e-learning videos: canvas-based, slide-based and recorded lectures. It also discussed the challenges each one of them poses to our technique.

3 RELATED WORK IN INTERACTION

This Chapter discusses techniques that have been proposed for extending interaction with videos - rather than Computer Vision techniques which make these extensions possible. The background necessary for the implementation will be discussed in subsequent chapters. To provide a systematic review of the area, we will divide our discussion into three main categories: non-linear video navigation, video annotation/summarization, and interaction with e-learning videos.

3.1 Non-linear Video Navigation

In this Section, we will discuss the research in non-linear navigation of videos. We call linear the navigation which consists of going through the video using a time slider. Non-linear navigation, then, includes all types of interaction where we do not go sequentially through time.

One of the first attempts to provide exploration that is not exclusively temporal is Cybercoaster (SATOU et al., 1999). The authors introduce the idea of using the trajectory of an object to navigate in the video: when the user moves an object to position p, the video is browsed to a frame where the object lies at p. Figure 3.1(c) shows a nice illustrations of this concept. The authors do not specify how to track the object, the main contribution is on the user interface design. Many works (KIMBER et al., July) (GOLDMAN et al., 2008) (DRAGICEVIC et al., 2008) (KARRER et al., 2008) were developed to put this design into practice, automatically tracking the objects.

Kimber et al. (KIMBER et al., July) introduced a system called Trailblazing, that allows this exploration by direct object manipulation in fixed-camera videos - specially for surveillance. They use a Gaussian Mixture Model (REYNOLDS, 2009) (discussed in detail on Appendix C) to model the background. Object detection and tracking are performed by background subtraction. They also demonstrate a mapping from the object trajectory in a multiple-camera environment to a trajectory in a floor plan. The user may navigate by direct manipulation either on the video or on this floor plan.

Dragicevic et al. (DRAGICEVIC et al., 2008) developed a method based on optical flow - tracking pixels instead of whole objects. The authors argue that optical flow is more reliable than object tracking. They also deal with background motion by estimating the most representative motion between two frames. This background motion is taken into account when calculating the relative motion of objects. Karrer et al. (KARRER et al., 2008) presented a similar technique, that used a slower, but even more reliable optical flow method.

Goldman et al. (GOLDMAN et al., 2008) introduced yet another system that provides video navigation by direct manipulation, extending the prior set of features to allow anno-

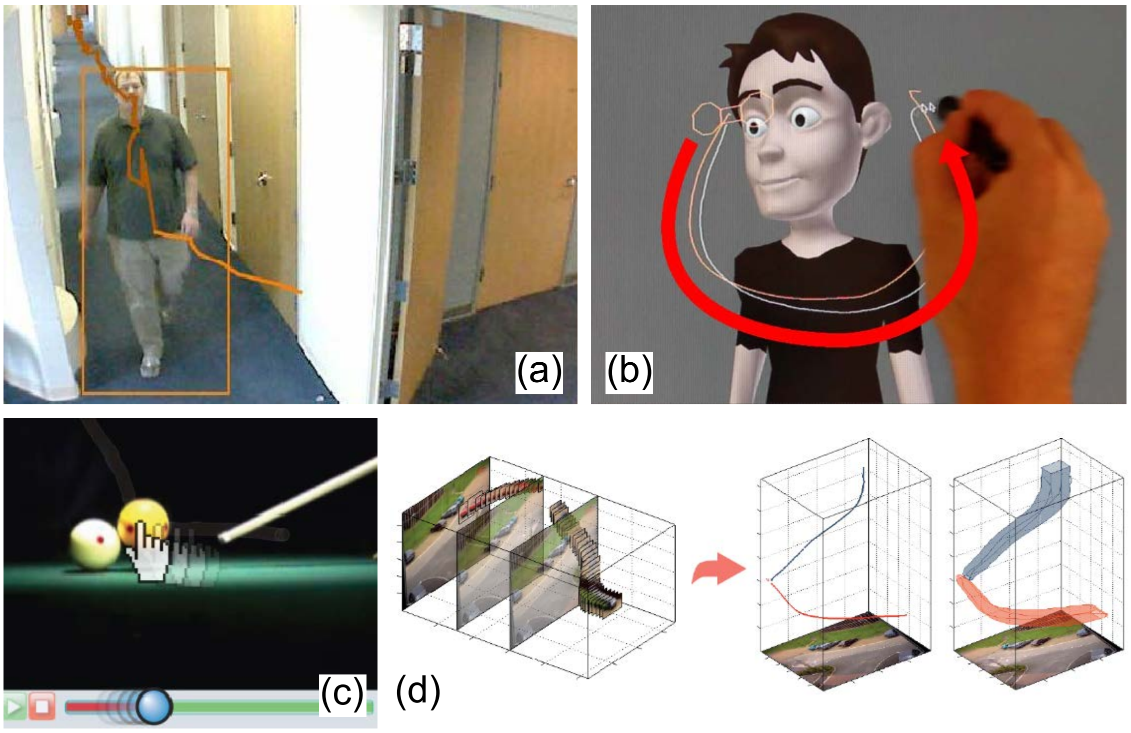


Figure 3.1: Related work in non-linear navigation of videos. (a) Trailblazing, first direct manipulation by object trajectory (KIMBER et al., July). (b) Dragimation, direct manipulation for computer animations (WALTHER-FRANKS et al., 2012). (c) Direct manipulation using per-pixel optical flow (DRAGICEVIC et al., 2008). (d) Recent work extending trajectory do 3D space-time representation (SHAH; NARAYANAN, 2013).

tations to be added to moving objects. They discuss the idea of hyperlinks on the video, which is closely related to our proposed navigation method. According to the authors, the idea of hyperlinks in videos has been introduced by Hypersoap (DAKSS et al., 1998), and the taxonomy defined by Smith et al. (SMITH; STOTTS; KUM, 2000). Differently from us, they use manually added hyperlinks, from annotations in the video to websites. We automatically create hyperlinks, from elements in a given frame to the moment when these elements were explained.

Walther-Franks et al. (WALTHER-FRANKS et al., 2012) introduced the concept of navigating in time by object manipulation in a different context. It explores the frames of a computer animation - not a video. Their work is aimed at helping animators finding the exact moment where an object was at a given position - they claim such navigations are unintuitive because computer animations are interpolated from key frames.

Very recently, Shah and Narayanan (SHAH; NARAYANAN, 2011) (SHAH; NARAYANAN, 2013) created a variation of the concept of direct manipulation by presenting the user with a 3D trajectory of the object. In videos, we can only detect whether the object has moved in the plane - their work uses the time as a third dimension. This results in a volumetric representation of the object trajectory that can be used for navigation (Figure 3.1(d)).

3.2 Video Summarization

A different topic closely related to ours is video summarization. This subject is extremely vast, since a summary can be of many different forms: a storyboard, a panorama, a set of images, or a 3D visualization of the video, among others. In our context, for example, we can see the final set of notes as providing a synopsis of the video. We will try to cover the main types of summarization in breadth. A more complete review can be found in Truong and Venkatesh (TRUONG; VENKATESH, 2007), though it does not cover the most recent research.

The first approaches usually focused on graphically representing the video in novel ways. One of the precursors of these visualizations is the work by Fels and Mase (FELS; MASE, 1999), which introduces the visualization of videos as 3D volumes, having time as a third dimension. Later, Daniel and Chen (DANIEL; CHEN, 2003) proposed some simple variations of the visualization as a cube, taking into account the differences in the frames to find important checkpoints of the video. Recently, Nguyen et al. (NGUYEN; NIU; LIU, 2012) introduced VideoSummagator, a new visualization for videos that also uses 3D volumes, in a more sophisticated way. They use computer vision techniques to detect dynamic voxels, which receive higher opacity values. An example of a video cube generated by VideoSummagator can be seen in Figure 3.2(a).

A different approach is the summarization of videos to a single image. Salient stills (TEODOSIO; BENDER, 2005) represents one of the first attempts in this area. The idea is to create a single image that preserves the salient features of each individual frame. An example of this technique can be seen in Figure 3.2(b). Assa et al. (ASSA; CASPI; COHEN-OR, 2005) considered the problem of selecting key frames or poses to generate a storyboard-like single frame. Goldman et al. (GOLDMAN et al., 2006) further extended this notion to create real single image storyboards - with the addition of arrows and frame outlines to better represent movement. The result can be seen in Figure 3.2(c).

More recently, two works on the creation of single images that represent videos have been introduced concurrently. Dynamic video narratives (CORREA; MA, 2010) and video tapestries (BARNES et al., 2010) both create summaries of long narratives by merging together a collection of still images, which are already formed by multiple frames. They refer to that process as summary of summaries. Examples can be seen in Figure 3.2(d)(e).

Finally, another possibility for summarization is the creation of video skims. Video skims (SMITH; KANADE, 1995) (SMITH; KANADE, 1998) (MA; ZHANG, 2002) (KANG et al., 2006) are shorter summary videos that represent the longer ones. The skims can be useful for navigation, but may require that the user observe the video for a long time. We will not review this type of summary in depth, since it is not closely related to our work.

3.3 Interaction with e-Learning Videos

The area where we believe our work is contributing most is the interaction with e-learning videos. While there is a huge accelaration in the availability of this type of resource, research dedicated to provide more effective interaction still has not followed.

Zhang et al. (ZHANG et al., 2006) is one of the first works to investigate interactivity in e-learning videos. They discuss a system based on a paradigm called learning by asking (LBA). The main idea is to have many short videos, each about one specific topic, that can be accessed randomly by the user - simulating the answer to one specific question. In the paper, they have one small video explaining each slide of a lecture. The user is able to navigate by slide. The authors report significant improvement when the students use this type of interactive e-learning videos, in comparison to sequential ones. One of the main restrictions is that it is necessary to generate and segment the videos in advance -

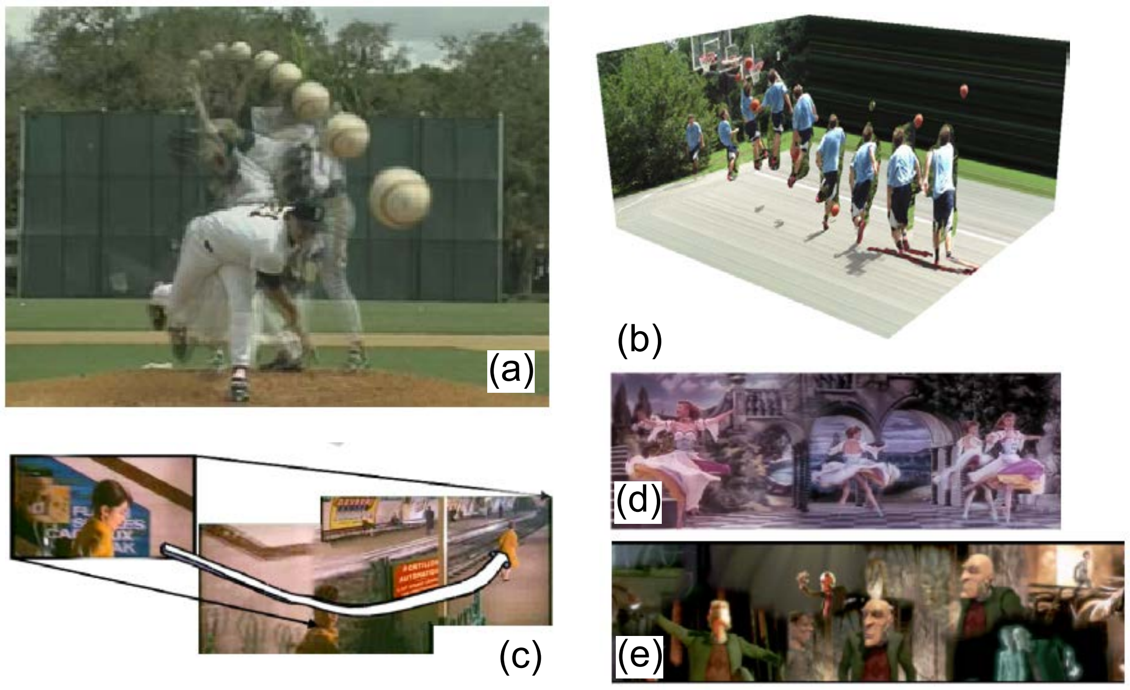


Figure 3.2: Related work in video annotation and summarization. (a) VideoSummagator (NGUYEN; NIU; LIU, 2012). (b) Salient stills, one of the precursors of single image video summarization (TEODOSIO; BENDER, 2005). (c) Goldman et al. (GOLDMAN et al., 2006) creates very expressive storyboards from videos. (d) Dynamic video narratives (CORREA; MA, 2010). (e) Video tapestry (BARNES et al., 2010).

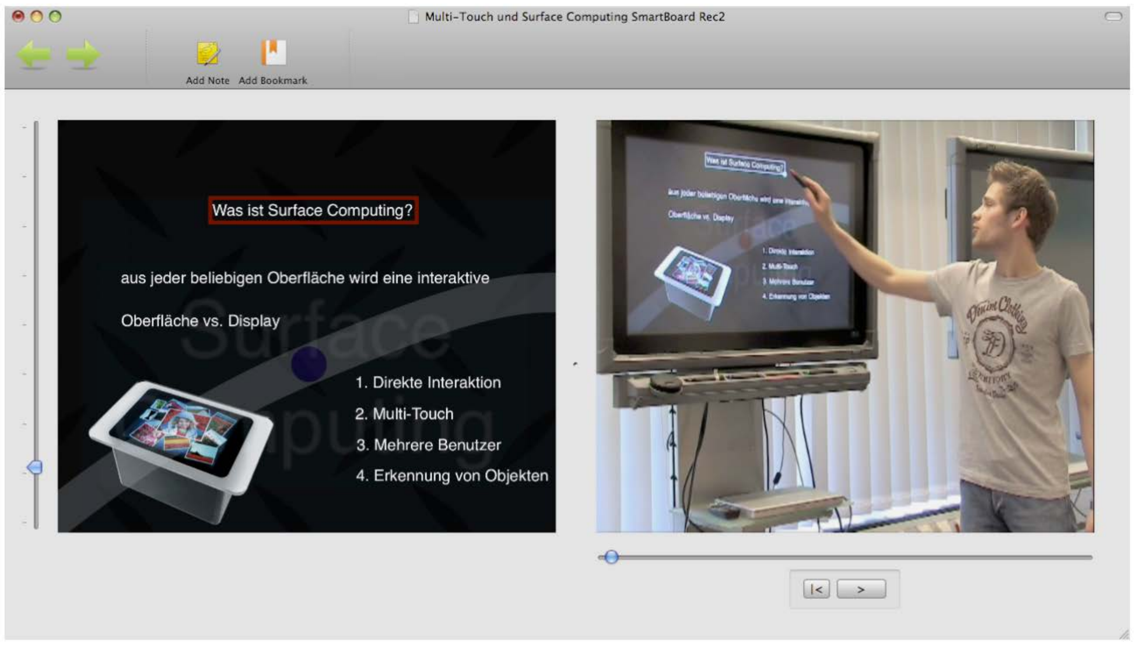

Multi-Touch und Surface Computing SmartBoard Rec2
Add Note
Add Bookmark
Was ist Surface Computing?
aus jeder beliebigen Oberfläche wird eine interaktive
Oberfläche vs. Display
1. Direkte Interaktion
2. Multi-Touch
3. Mehrere Benutzer
4. Erkennung von Objekten
|<
>

Figure 3.3: The interface of DragonFly (CORSTEN, 2010). Left: the interface for navigation, showing a planar presentation pre-generated with a tool. Right: the video of the lecture using the presentation.

we cannot apply this technique to existing videos without pre-processing.

One of the most recent works in interaction with video lectures, and certainly one of the most closely related to ours, is DragonFly (CORSTEN, 2009) (CORSTEN, 2010). DragonFly is a system to spatially navigate through video lectures of slide presentations - the interface can be seen in Figure 3.3. The idea is that the video lecture uses a particular type of slide presentation (HOLMAN et al., 2006) that already has a sort of planar representation. This map is then used to provide a spatial slider in addition to the regular time slider.

The main limitation of the system is that it only works for videos of lectures that use this specific type of presentation. That limits the application of this tool to a very narrow portion of the existing e-learning videos. Our technique, on the other hand, can not only handle existing videos but is also general enough to be easily customizable to different categories of videos we might not have seen yet. Also, the spatial navigation provided by DragonFly is not based on the true spatial relations between frame but in artificially created relations in a map. Finally, the users from DragonFly reportedly complained about the lack of direct manipulation on the video, which is the base of our system.

3.4 Summary

In this Chapter, we discussed the most relevant research related to our work. We reviewed three sub-areas: **non-linear navigation**, that deals with navigation of videos guided by elements other than time, **video summarization** and **interaction with e-learning videos**.

4 RELATED WORK IN COMPUTER VISION

One of the main goals of our framework is to replace the usual linear navigation in time by a content-based navigation. In most of the videos we are interested in, there is implicit information of the position of each frame in relation to others. In the first part of this Chapter, we will discuss the problem of aligning video frames to a common coordinate system. In the second part, we will review work related to the object tracking - used to solve occlusion problems when the instruction is in front of the board.

4.1 Video Registration

The applications of image and video registration are many. The most obvious ones are the automatic creation of panoramas out of a set of images of the same scene (BROWN; LOWE, 2007) and the augmentation of videos to be panoramic (AGARWALA et al., 2005) (HERMANS et al., 2008). Less immediate applications include new approaches to navigate through many photographs of a scene (SNAVELY; SEITZ; SZELISKI, 2006), interfaces for re-taking a photograph from a given viewpoint (BAE; AGARWALA; DURAND, 2010), alignment of videos for various effects (SAND; TELLER, 2004), among others.

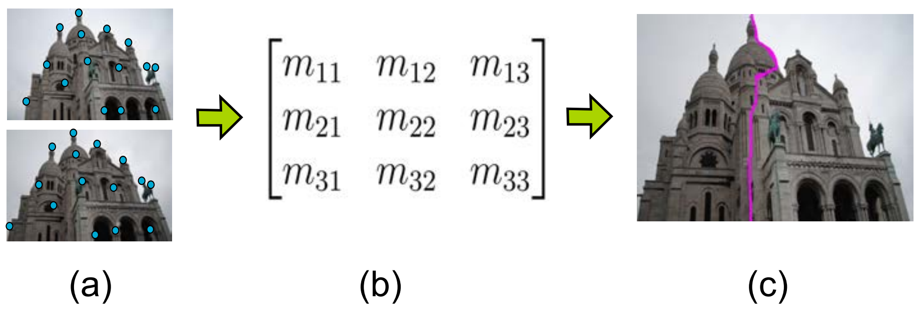

$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$
(a)
(b)
(c)

Figure 4.1: Common pipeline to image and video registration applications. (a) Feature detection and matching. (b) Transformation estimation between two images. (c) Image stitching with seam optimization.

Image registration applications usually go through a common pipeline. Suppose we have a sequence of images of a scene - partially overlapping - that we want to put in the same coordinate system. For simplicity, we will consider registration between pairs of images. First, we need to identify parts of the image that are distinguishable enough to be matched reliably. We do this separately for each image. Then, we find the most similar pairs of feature points and match them. These first steps are called feature detection and matching and are illustrated in Figure 4.1(a).

Once we have a set of match candidates, we want to find a transformation T from the first set of points to the other. Two issues have to be observed in this step: first, many of the candidate matches might be outliers, due to mismatched features; second, even features that are correctly matched might have small errors due to lack of precision in feature detection. Taking that into account, we first estimate the transformation using RANSAC (FISCHLER; BOLLES, 1981), which is robust to outliers. Once we have an initial estimation of T and the set of points we consider as inliers, we use every point in this set to refine the result.

Finally, even perfectly aligned frames might have inconsistencies - a moving object, a change in illumination, etc. To create a new image from the two existing ones, we need to stitch the images together in a way that artifacts are not noticeable. As a first approximation, cutting images around the edges usually leads to more visually plausible results. The complete pipeline can be seen in Figure 4.1. In the next two sections, we will discuss the first two steps of the image registration pipeline in more detail. While we implemented some image composition techniques during the development of our interface, they were not used in the final version of the system. To avoid confusion, we moved this discussion to Appendix A.

4.2 Feature Detection and Matching

Feature detection and matching are an essential component of video registration applications. Consider two frames of a video and the task of creating a panorama from them. While there is much coherence in the two images, the information about how they would fit together is not explicit. The similarity of a part of the first image with a part of the second image needs to be formalized in a way that a computer can understand it. We will now review the detection and matching of points and patches. We will not discuss the detection of lines at this point because the difficulty in the detection of such high-level features makes them unreliable for registration (HESS; FERN, 2007). Moreover, points and regions of interest provide the most generally applicable algorithms (SZELISKI, 2010). The approach we will discuss finds the most prominent features of each image and then tries to match the sets.

We will organize the discussion in this Section in three main steps:

- **Feature detection:** The first step we are concerned with is finding which parts of the image make the most distinctive features, in order to avoid ambiguities;
- **Feature description:** Once we found good points and patches, we want to describe them in a way that is memory efficient and, more importantly, invariant to the transformations a feature may suffer when projected;
- **Feature matching:** We need to compare features to find which ones correspond to the same entity in a scene. We would be interested in estimating how close our matched features are from each other, *i.e.*, estimate a level of confidence for our match.

4.2.1 Feature detection

This first question that comes up when confronted with the task of selecting a set of features from an image is: what makes a good feature? Intuitively, one can immediately answer that the most distinctive patches should be selected - in Figure 4.2, patch (b) makes a characteristic feature, while (a) and (c) would be much harder to position exactly in the second image, even for a human observer.

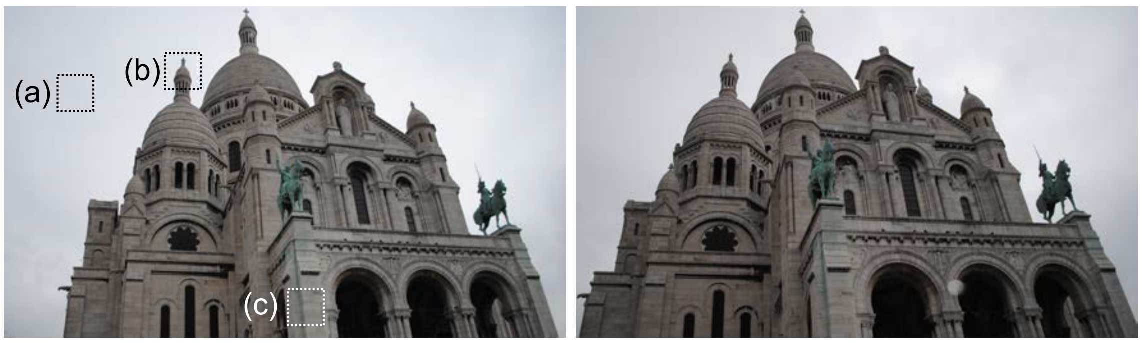


Figure 4.2: Different patches of the image.
We can see (b) is a good feature, while (a) and (c) suffer from ambiguity.

Let us make this intuition more clear. Given a patch p in an image, the simplest algorithm to find its correspondent patch in the other image is to look at all possible patches and take the one most similar to p. One way to think of that in a more formal way is trying to minimize the sum of squared differences between them. This operations can be described by the following equation, where I_0 and I_1 are the images being compared, and u is a displacement vector (representing the position of the patch in I_1).

$$Error(u) = \sum [I_1(x_i + u) - I_0(x_i)]^2 \tag{4.1}$$

Now observe again Figure 4.2. If we move the patch (a) a little bit, there is almost no change in the color. That means that the equation above would have very similar results when compared to any patch in the region around (a). Also, for patch (c), if we move along the edge, there would be only a small variation of the color of the patch. In contrast, moving patch (b) would generate a patch that is very different from it. To avoid ambiguity, that is exactly what we want - that the patch cannot be mismatched without increasing the error function. Now, in our description we use the idea of matching p to patches in a different image - but we can easily remove that dependency. If p cannot be mismatched to neighbor patches, it is distinctive enough to avoid ambiguities. The exact formula can be seen below and is called **autocorrelation function**.

$$AutoCorrelation(u) = \sum [I_0(x_i + u) - I_0(x_i)]^2 \tag{4.2}$$

Figure 4.3 shows the autocorrelation results for three distinct cases (not the same as the patches shown in Figure 4.2). The error is plotted in terms of the displacement. We can see that the autocorrelation matrix on the left generates a distinctive minimum, while patch the matrix at the center generates a minimum line and the matrix on the right generates a function with many minima. The phenomenon illustrated in the matrix in the center is called aperture problem, and is caused by ambiguity along a single direction (HORN; SCHUNCK, 1980).

We can approximate the function using the first terms of a Taylor Series expansion, to generate a matrix that will then be used to define how distinctive a feature is. Using $\nabla I_0(x_i) = (\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y})(x_i)$ as the gradient of an image at point x_i, we can approximate the value of the image $I_0(x_i+u)$ as $I_0(x_i)+\nabla I_0(x_i)u$. Note that u is the displacement vector,

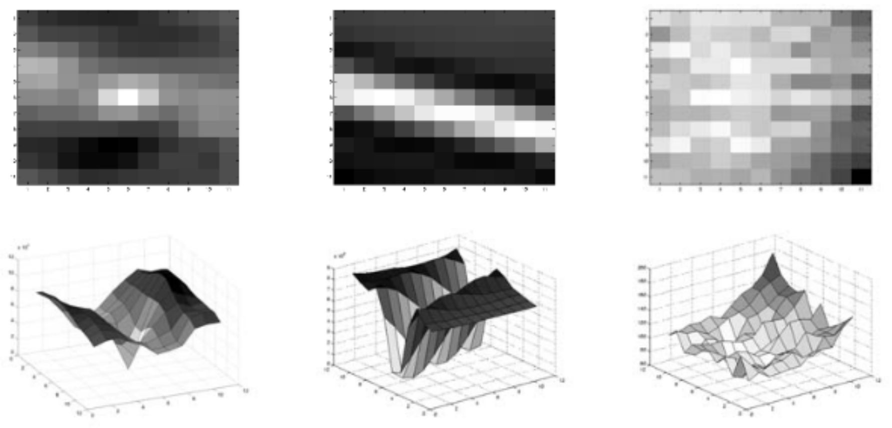


Figure 4.3: Three types of features and the respective autocorrelation values. (left) a good feature with a distincted minimum. (center) a feature suffering from aperture (HORN; SCHUNCK, 1980). (right) a very bad feature for which many points in the neighborhood are minima. Figure adapted from (SZELISKI, 2010).

and does not vary with the image. This rewriting allows some interesting simplifications, as shown in Equation 4.3.

$$\begin{aligned} AutoCorrelation(u) &= \sum [I_0(x_i + u) - I_0(x_i)]^2 && (4.3) \\ &\approx \sum [I_0(x_i) + \nabla I_0(x_i)u - I_0(x_i)]^2 && (4.4) \\ &= \sum [\nabla I_0(x_i)u]^2 && (4.5) \\ &= \sum u^T A u && (4.6) \\ & && (4.7) \end{aligned}$$

The matrix A is called the **autocorrelation matrix** and its properties can be used to decide whether a feature is distinct or not. It is defined as follows (SZELISKI, 2010).

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \tag{4.8}$$

How the properties of the autocorrelation matrix are used to determine whether a feature should be used or not depends on the type of detection we do. We will discuss some examples of the literature.

Shi and Tomasi: One of the first analysis of the autocorrelation matrix is due to Shi and Tomasi (SHI; TOMASI, 1994). The eigenvectors of a correlation matrix indicate the direction of maximum variance from a dataset and the respective eigenvalues show the magnitude of the variance of the set along that direction. If the correlation matrix of a feature has small eigenvalues then no direction can represent a high variability, which means the window is composed by a roughly constant intensity. Similarly, a matrix with a large eigenvalue and a small one represents an image that has high variability only in one direction, probably an edge. Shi and Tomasi (SHI; TOMASI, 1994) suggest that the good features to track are given by an autocorrelation matrix where both eigenvalues are above a given threshold, which would represent a corner, a salt-and-pepper texture, or other patterns that can be reliably tracked.

Förstner-Harris: The minimum eingenvalue is not the only possibility for analyzing the properties of the autocorrelation matrix. A simpler approach, described independently by Harris and Stephens (HARRIS; STEPHENS, 1988) and Förstner (FöRSTNER, 1994), is to use the determinant and the trace of the matrix A.

$$det(A) + \alpha \times trace(A), \alpha = 0.06 \tag{4.9}$$

Note that the determinant of A is equivalent to the product of the eigenvalues of A, $\lambda_0\lambda_1$, while the trace is equal to their sum $\lambda_0 + \lambda_1$. Alternative functions from the eigenvalues have been been proposed such as using $\lambda_0 + \alpha\lambda_1$ (TRIGGS, 2004) and $det(A)/trace(A)$ (BROWN; SZELISKI; WINDER, 2005).

SIFT: In some cases, it is important that an object can be identified despite the differences in scales between the two images. Our detectors should find features that are stable with respect to both scale and location. The most important work in scale invariant features is due to Lowe (LOWE, 1999). We will discuss this work in more depth than others - both because of its relevance and because it was used in our implementation.

Before describing this type of feature detector, we should define a representation of the image that considers different scalings. The scale space theory is a framework for handling images at different scales, by representing one image as function of three parameters - the usual x and y denoting position, and one additional scaling parameter σ. We consider that the convolution of the image with Gaussian kernels of variable size supresses the fine scale details giving many levels of coarseness - that could be interpreted as smaller versions of the image. Let us denote $L(x, y, \sigma)$ as the scale-space representation, then $L(x, y, \sigma) = I(x, y) * G(x, y, \sigma)$, where I is the image, G is a Gaussian filter and σ is the kernel size. The symbol $*$ denotes a convolution. We can see some levels of the scale space of an image in Figure 4.4.

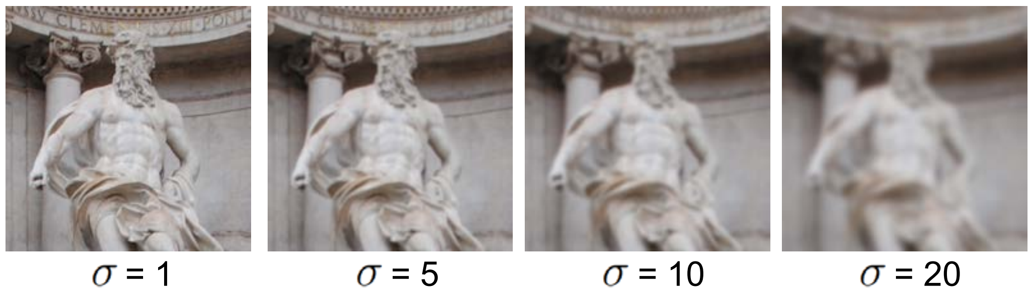


Figure 4.4: Scale-space. Convolution with Gaussian kernels of different sizes yield the multiple scale representation.

The idea is to search for features that are stable across the whole scale space. The technique searches for extrema using differences-of-Gaussians, that can be written as the difference between two images at different scales, as seen in the equation below. k is a multiplier that denotes the next scaling level.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) & (4.10) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) & (4.11) \end{aligned}$$

To detect local extrema, we compare a pixel with its 3D neighborhood. The pixel is only selected as a maxima or minima if it is bigger (or smaller) than its 8 neighbors on the image, the 9 neighbors in the next level of the scale space and the 9 neighbors in the previous level of the scale space. It is important to emphasize that this detection is performed using differences-of-Gaussians. Each point p that is a feature candidate will have a position defined by (x, y, σ). From this initial point, we can fit a function, with the origin located in p, to determine the exact location of maximum (which could be an interpolated location - see Figure 4.5).

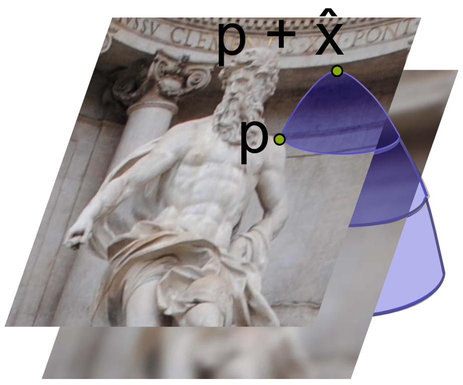


Figure 4.5: Fit a function to find maximum value. Note that $\hat{x}$ can lie between samples in any dimensions, x, y or σ.

The function can be approximated by a Taylor series expansion - up to quadratic terms - see Equation 4.12. In this equation, x is the vector of parameters defined by the offset (x, y, σ) from the origin - the sampled point. The position of the maximum $\hat{x}$ can be found by taking the derivative of equation 4.12 relative to x and setting it to zero. The final formula for $\hat{x}$ is Equation 4.13.

$$D(x) = D + \frac{\partial D^T}{\partial x} + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2} \tag{4.12}$$

$$\hat{x} = -(\frac{\partial^2 D}{\partial x^2})^{-1}\frac{\partial D^T}{\partial x} \tag{4.13}$$

The term $\frac{\partial^2 D}{\partial x^2}$ is the Hessian matrix. Both the Hessian and $\frac{\partial D}{\partial x}$ can be approximated by finite differences. Note that x here is a vector formed by the original (x, y, σ). The Hessian, then, is a 3x3 matrix and the Equation 4.13 results in a 3x3 linear system. Once we have the offset $\hat{x}$, we can add it to the sample point p.

Again, it is important to take care of edges - which are detected as peaks by the algorithm above, and would cause ambiguities along one direction. To remove edges, we can use the same strategy as in Harris detector - we simply calculate the autocorrelation matrix in the scale space of the sample point. Following the development by Harris (HARRIS; STEPHENS, 1988), we can use the singular values of A to determine the variability along the two main directions. Also, we can once again avoid calculating the eigenvectors by using Equation 4.9.

4.2.2 Feature description

Once we have decided which patches of the image would generate good features, we need to match them. A simple approach would be minimizing the sum of squared differences. Unfortunately, this would only work if the transformation applied to the patch was a translation. In most cases, the patch will undergo arbitrary affine transformations that will require more sophisticated methods for comparison. In this Section, we discuss some ideas on how to describe our patch information to be invariant to some transformations.

MOPS: The multi-scale oriented patches (BROWN; SZELISKI; WINDER, 2005) are mostly used in applications that do not require invariance for transformations, due to the simplicity of implementation. It consists of sampling a lower frequency version of the image around the feature. Using a coarser version makes it more robust to inaccuracies in feature detection. Patch intensities are re-scaled such that their mean is zero and the variance is 1 to compensate for photometric changes, like exposure variations.

SIFT: The descriptors for SIFT features are formed by computing the gradient at each pixel in a 16x16 window around the feature point - the feature point scale is used to determine the level of Gaussian blur - see Section 4.2.1. The magnitude of each gradient is weighted by a Gaussian centered at the pixel, to give less emphasis to gradients far from the center. The 16x16 window is divided into sixteen 4x4 regions, and the gradient of each of these regions is represented by a histogram with 8 bins. Each pixel in the region is added to this histogram to form the final descriptor. The 128 resulting values (the values in each of the 8 bins, in each of the 16 regions, 16x8 = 128) form the SIFT descriptor. A schematic drawing of this process, using a 8x8 window grouped in 4 regions, can be seen in Figure 4.6.

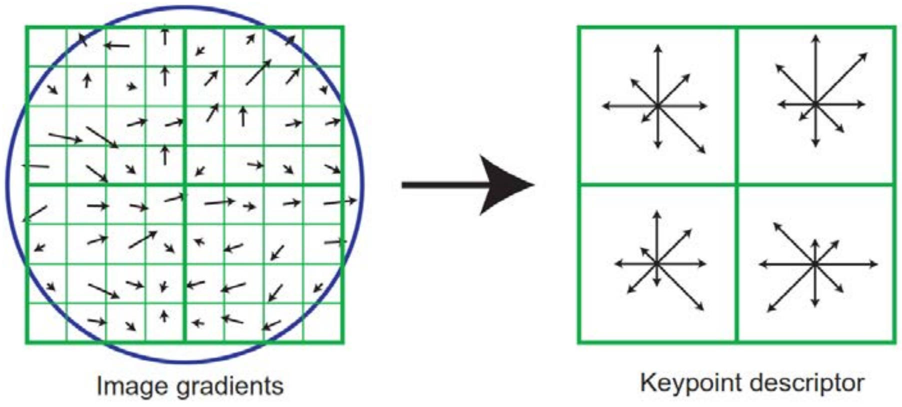


Figure 4.6: Sift descriptor. On the left, we can see each pixel contributing with a gradient for the final descriptor. The blue circle indicates the Gaussian weighting. On the right, we can see the grouping of the pixels into regions. Taken from (LOWE, 1999).

GLOH: The Gradient Location-Orientation Histogram (MIKOLAJCZYK; SCHMID, 2005) is basically an extension of SIFT that uses polar bins instead of square ones. They divide the space into 3 radial bins and 6 angular bins - with one additional bin for radius = 0.

4.2.3 Feature matching

There are two important steps when matching features. The first one is to define a matching strategy; the second one to combine data structures and algorithms in order to efficiently perform the matching evaluations.

One of the simplest matching strategies that can be adopted is to immediately reject matches that are further away from each other than a threshold t, using Euclidean distance. In this case, we must of course observe that the threshold is consistent with our expected camera motion, to avoid false positives and false negatives. The problem with this strategy is that the threshold depends on each case and is difficult to optimize. A different approach is to use nearest neighbors in feature space. Since some features would not have matches, a threshold approach is still used to avoid extreme false positives.

The naïve approach to comparing features is $O(N^2)$, comparing all features against each other. The data structures used to improve the performance of this search are numerous. Muja and Lowe (MUJA; LOWE, 2009) reviewed many of the existent data structures for the problem. Their benchmarks showed that multiple randomized kd-trees yield the best results for many cases.

4.3 Homography Estimation

In this Section, we will discuss the problem of estimating the transformation from one image into another, given point correspondences between them. At first, we will study some basic definitions necessary to understand this transformation and its properties.

4.3.1 Projective Plane and Homogeneous Representation

The first important concept in understanding camera transformations is the projective space. The projective space is, informally, the representation of what a camera can see. In that sense, things that are equivalent for the camera are equivalent in this space. The only information a camera uses when forming an image is the direction and intensity of the incoming light rays. It cannot distinguish between two different points lying on the same ray. Therefore, a reasonable representation for the projective space, from now on referred to as P^2, is a set of rays in R^3, as can be seen in Figure 4.7. The stablished convention is to use the set of rays that go through the origin.

All points along a ray in P^2 are equivalent. Algebraically, we represent this by the use of homogeneous coordinates, *i.e.*, a point p = (x, y)^T is represented by p = (x, y, 1)^T or any multiple of it. To find the 2D (inhomogeneous) coordinates of the point, we simply divide all coordinates by the last one. The vector v = (0, 0, 0)^T does not belong to the projective space.

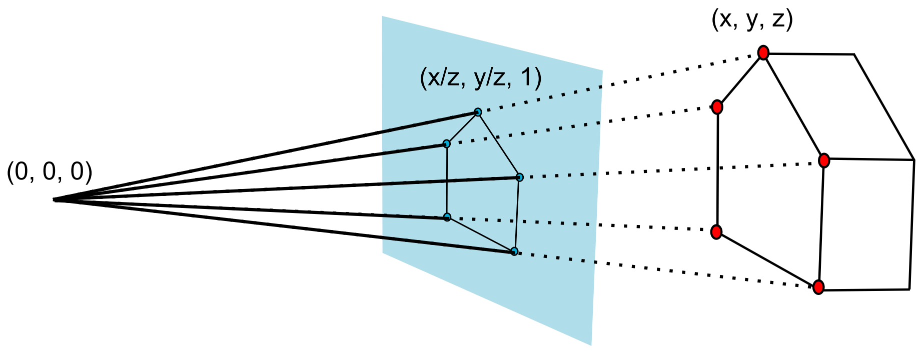


Figure 4.7: Representation of projective plane P^2 as rays in R^3.

4.3.2 Projective Transformations

A projective transformation is a linear inversible mapping H from a point p in P^2 to another point p' in P^2 that preserves collinearity. If p_1 and p_2 lie on the same line, Hp_1 and Hp_2 should also be collinear. A common name for a projective transformation that will be used throughout this text is *homography*.

The matrix H is a homogeneous matrix, *i.e.* any multiplication of H by a scalar is equivalent to H. In fact, the homography is a simple linear transformation in R^3, and the ambiguity results form the use of projective space.

4.3.3 The Direct Linear Transformation (DLT) Algorithm

We consider a set of point correspondences $x_i \leftrightarrow x'_i$ in two images. Our task is to compute an arbitrary 3 $\times$ 3 matrix H, such that $x'_i = Hx_i$. The 2D points will be treated as homogeneous vectors with $w = 1$.

The simplest way to compute a homography is to set up the correspondences as contraints and solve a linear system. It is important to note that the points are in the projective space, *i.e.* they have homogeneous coordinates. That means x'_i is not necessarily equal to Hx_i, as they may differ by a scaling factor. Thus, our constraint has to be the parallelism of the two rays, not their equality. The parallelism of these two rays in R^3 can be described by setting the cross product to zero.

To make our calculations more intuitive, our first step will be to write the product Hx_i as the dot product of the rows of H and the vector x_i. The n-th row of the matrix will be denoted H^j.

$$Hx = \begin{pmatrix} H^1 \cdot x \\ H^2 \cdot x \\ H^3 \cdot x \end{pmatrix} \tag{4.14}$$

Taking the cross product of two 2D vectors a and b is equivalent to taking the determinant of a matrix where the first row is formed by the unit vector i, j and k, the second row is formed by the components of vector a and the third row is formed by the components of vector b.

$$x' \times Hx = det \begin{vmatrix} i & j & k \\ x'_1 & x'_2 & x'_3 \\ H^1 \cdot x & H^2 \cdot x & H^3 \cdot x \end{vmatrix} = \begin{pmatrix} (H^3 \cdot x)x'_2 - (H^2 \cdot x)x'_3 \\ (H^1 \cdot x)x'_3 - (H^3 \cdot x)x'_1 \\ (H^2 \cdot x)x'_1 - (H^1 \cdot x)x'_2 \end{pmatrix} = 0 \tag{4.15}$$

We want to isolate the elements of H, since these are our unknowns. Using the distributive property of the dot product ($(A \cdot B)k = (A \cdot Bk)$), we can write the matrix above as follows.

$$\begin{bmatrix} 0 & 0 & 0 & -x_1x'_3 & -x_2x'_3 & -x_3x'_3 & x_1x'_2 & x_2x'_2 & x_3x'_2 \\ x_1x'_3 & x_2x'_3 & x_3x'_3 & 0 & 0 & 0 & -x_1x'_3 & -x_1x'_1 & -x_3x'_1 \\ x_1x'_1 & x_2x'_1 & x_3x'_1 & -x_1x'_2 & -x_1x'_2 & -x_3x'_2 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} H^{11} \\ H^{12} \\ H^{13} \\ H^{21} \\ H^{22} \\ H^{23} \\ H^{31} \\ H^{32} \\ H^{33} \end{pmatrix} = 0 \tag{4.16}$$

Note that, since we are dealing with homogeneous coordinates, a correspondence between two points only has 2 degrees of freedom (the ratios between x and w and y and w). Indeed, the last constraint does not add any information to the linear system and is usually discarded (HARTLEY; ZISSERMAN, 2004). Now, each point provides us with two constraints, and the matrix H has 8 degrees of freedom (9 matrix elements of the matrix less 1 for the scaling factor), so we need 4 point correspondences to determine the exact transformation between two images. The resultant vector will be determined only up to scale, so an additional constraint has to be used. Two usual examples are setting the last element or the norm of the vector to 1.

Normalization. A point p = (200, 200)^T is not uncommon when considering image coordinates. This would lead to entries of magnitude 4×10^4 in the matrix H - which is bad conditioned and can lead to unstable results. The common way to solve this - which should not be considered as an optimization, but as part of the method - is to normalize point coordinates x and x' before applying DLT. The normalization should be implemented as follows (SZELISKI, 2010):

- Translate the points so that their centroid is at the origin.
- Scale the point coordinates so that the average distance to centroid is $\sqrt{2}$.

This normalization should be applied independently to both images being matched. Also, the homography must be denormalized once DLT is over. Suppose T_1 is the transformation applied to the first image and T_2 is the transformation applied to the second one. The final homography will be $H = T'^{-1}\hat{H}T$, where $\hat{H}$ is the normalized homography.

Minimal Solutions and Over-Determined Systems: For every correspondence $x_i \leftrightarrow x'_i$, we can add two constraints on the values of the matrix H. If we have exactly four correspondences, this is called a minimal solution. If more than four correspondences are given, the system will be over-determined. In the presence of noise, an over-determined system will not have an exact solution $Ah = 0$. We must therefore solve this system of equations in a least-squares sense. Since we are ideally looking for $Ah = 0$, a reasonable choice of cost function to be minimized is $||Ah||$. It can be shown that h will be the unit singular vector correspondent to the smallest singular value of A. This can be easily calculated using the Singular Value Decomposition (SVD) (STRANG, 2006), available in many linear algebra libraries.

Degenerate Configurations: When estimating a homography from 2D point correspondences, it is important to verify that our configuration is not degenerate. A degenerate configuration will lead to a homography that is not significant. For example, let us consider a minimal solution from 4 points, where 3 of the points, x_1, x_2 and x_3 are collinear. If the correspondent points (let us call them x'_1, x'_2 and x'_3) are also collinear, we can infer that the homography is not sufficiently constrained, while if they are not collinear there cannot be a homography H mapping $x_i \rightarrow x'_i$, since homographies preserve collinearity.

Let l be the line where the 3 collinear points lie, such that $l^T x_i = 0$. Now suppose we have a matrix $H^* = x_4 l^T$ of rank 1 (all matrices derived from the outer product of two vectors have rank 1). It is easily verified that the matrix H^* satisfies the cross product condition for x_i, $i = 1, 2, 3$, since $H^* x_i = (x_4 l^T) x_i = x_4(l^T x_i) = 0$ and the cross product of a vector with zero is always zero. For x_4, we have $H^* x_4 = (x_4 l^T) x_4 = k x_4$, which would also result in a zero cross product.

The problem with this solution is that the rank of matrix H^* is 1, which means it is not a projective transformation. If H^* is the only solution to the system, there is no homography mapping x_i to x'_i. If there are other solutions, then the system is clearly not constrained enough.

4.3.4 Robust Estimation using RANSAC

When estimating a homography the way described in Section 4.3.3, we assume a Gaussian distribution for the errors in point correspondences. Under this assumption, we can minimize a cost function using least squares to achieve a satisfactory result. In practice, however, feature matching will often generate wrong matches. These points will be outliers in the Gaussian distribution. If we take these outliers into account in a normal least-squares minimization, they may severely compromise the result. The problem is demonstrated in Figure 4.8(a) - a single outlier can lead to a bad estimation in an otherwise perfect measurement.

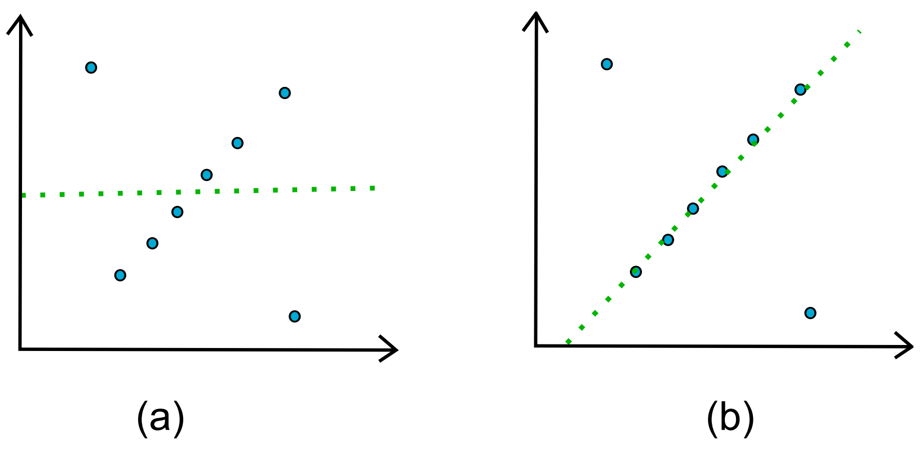


Figure 4.8: Least-squares problem when dealing with outliers. (a) Least-squares result (b) Result when outliers are removed

The goal of robust estimation is to make good estimations in the presence of a large proportion of outliers. We will discuss in detail one of the most used robust estimation algorithms, the RANdom SAmple Consensus (RANSAC) (FISCHLER; BOLLES, 1981).

The idea behind RANSAC is a very simple, yet powerful one. Suppose we are estimating a homography and therefore need 4 points for a minimal solution. We select 4 points randomly from the set of all points and estimate the homography based on these 4 points. Then we go through all point correspondences and see whether they would agree with this homography, up to a small threshold t. The number of inliers is the *support* of the given homography in this set of point correspondences. We repeat this procedure a number of times to find the homography with the biggest support, which is then considered to be the robust fit.

One of the problems with RANSAC which can be seen right away is that, as an iterative algorithm, it is not guaranteed to return the best possible solution. Three main parameters must be optimized to give the best possible results: the threshold t, the number of iterations and the minimum size of a consensus set.

The distance threshold: The maximum distance that a feature can be from its correspondent to be considered an inlier is one of the parameters to be chosen when applying RANSAC. It is usually chosen empirically. If the error distribution is known, the threshold t can be optimized such that it will select inliers with high probability. The problem with misestimating the threshold is that, if the value is too high, any fit in a range will be considered good enough, if the value is too low there will not be enough inliers even for the right fit. These problems are demonstrated in Figure 4.9. In my experience, the best results were achieved by starting with a small threshold t and go up until there are enough inliers (until a certain limit).

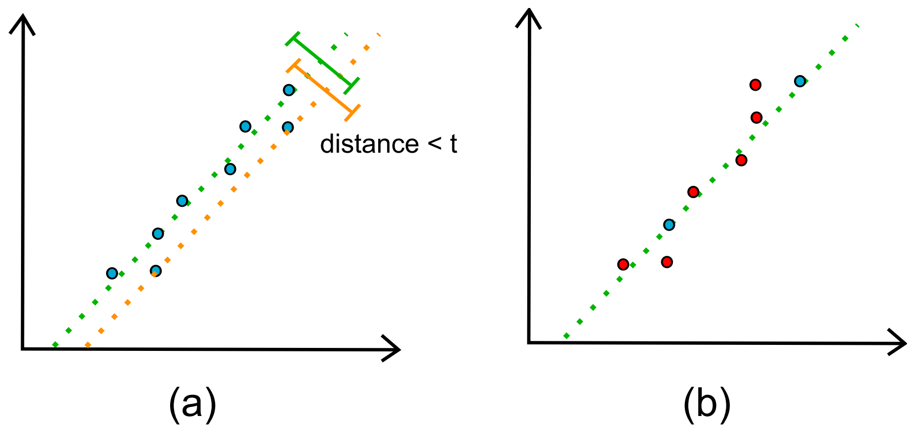


Figure 4.9: Threshold problem in RANSAC. (a) Threshold is too high - many solutions are considered optimal (b) Threshold is too low - not enough inliers

The number of iterations: For N matched points, the total number of homography estimations to test all possibilities would be $\frac{N!}{(N-4)!4!}$, which is $O(N^4)$. This calculation is often computationally infeasible. Instead, we try to ensure that at least one sample consists of inliers only with a high probability p. Suppose w is the probability that a point is an inlier. Then, we need to select at least M samples with s points each such that $(1-w^s)^M = 1-p$. If we set $\epsilon = 1-w$ to be the probability that a point is an outlier, we can rewrite this equation as

$$\begin{aligned} M &= log_{(1-w^s)}(1-p) \\ &= \frac{log_2(1-p)}{log_2(1-w^s)} \\ &= \frac{log_2(1-p)}{log_2(1-(1-\epsilon)^s)} \end{aligned} \tag{4.17}$$

The probability p is usually chosen to be 0.99, while ϵ would normally assume conservative values depending on the model being fitted. A different approach is to estimate the number of samples adaptively - at each iteration we update M using Equation 4.17, with probability $\epsilon = (1 - \frac{\#inliers}{\#points})$.

The minimum support: Since we already used a probability w that the point is an inlier, we can assume that the support of the homography is good enough when the number of inliers is close to the number of expected inliers, *i.e.*, $\#inliers \approx \#points * w$.

4.4 Object Tracking

This part of the Chapter reviews Computer Vision techniques dedicated to address one peculiar situation found in e-learning videos, that has to be solved before we can apply the techniques described in the previous chapter. One main problems was found when dealing with conventional recorded lectures: and detecting the instructor, whose features are not part of the important content of the lecture, and as such, should not be taken into consideration during most of our pipeline. Note that the occlusion of the content by the instructor is not a problem, because as long as he is the center of the frame - which is most often the case - the panorama will be formed correctly (for an example, see Figure 6.2).

We are interested in detecting the instructor, such that features that are not part of the important content are not take into consideration in the rest of the pipeline. To be able to remove these SIFT points precisely, we need to know the whole silhouette of the object. For simplicity, let us consider we already have a segmented object in the first frame, and are only interested in tracking this contour through the video. Yilmaz et al. (YILMAZ; JAVED; SHAH, 2006) name this problem *contour evolution*.

A different area, called video matting, substantially overlaps with contour evolution. Matting refers to the problem of estimating the foreground of a video, usually from little user interaction such as strokes. If we consider a unique object as the foreground, the result is the same. For a review of the state-of-art on matting, see Wang and Cohen (WANG; COHEN, 2008).

4.4.1 State space models

One way to think about contour evolution is to define a formal description of the characteristics of the object and track these characteristics to update the contour. Features that can be used to model the object are, for example, the shape of the contour and the color distribution. That is the state of the object. It is updated trying to maximize the a posteriori probability of the contour.

Isard and Blake (ISARD; BLAKE, 1998) maximize this probability using a particle filter. Given a set of observations z, the state x of the object is estimated using Bayes' rule:

$$p(x \mid z) = \frac{p(z \mid x)p(x)}{p(z)} \tag{4.18}$$

Note the $p(z)$ is equal to one, since this is the probability of an observation that has already been done. The authors define the object state using a spline to model shape parameters and an affine motion to model the dynamics.

Chen et al. (CHEN; RUI; HUANG, 2001) proposed a silhouette tracking where we use a Hidden Markov Model to propagate the state. To adapt the HMM to be applied in the image instead of in time, we note that the contour we are tracking is a 1D function. We can sample around the countour setting each sampled point to be a node of the HMM. This parametrization can be seen in Figure 4.10.

4.4.2 Roto Brush

Roto brush (BAI et al., 2009) was introduced in 2009 as a significant advancement of the state-of-art in the video matting community. We will review this technique in more detail, since we use it in our implementation. It could be classified as a state space model.

The algorithm is initialized with an accurate mask separating background and foreground. Given this initial mask, a set of overlapping windows is created by uniformly sampling along the contour (Figure 4.11(a)). Each of these windows will have a local classifier that will define a probability of the pixel belonging to the foreground. Each classifier consists of a color model M_c, a color model confidence f_c, and a shape model M_s. Let us discuss these three elements separately.

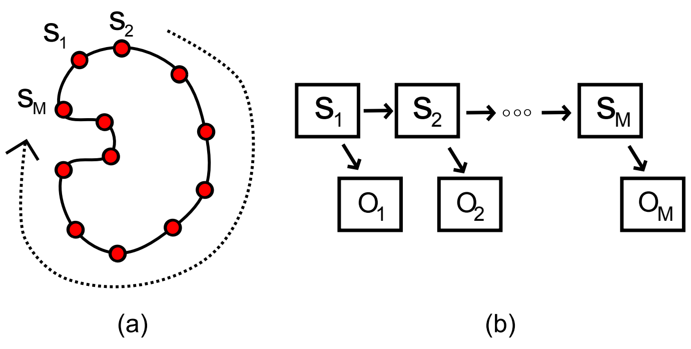


Figure 4.10: Countour evolution using Hidden Markov Models. (a) Sampled contour. (b) HMM representation.

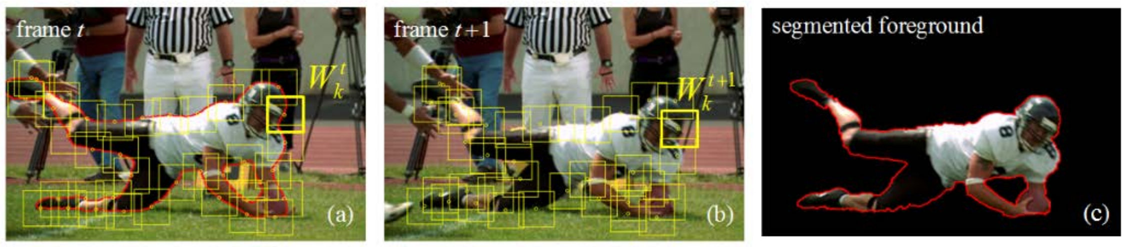


Figure 4.11: RotoBrush - Figure adapted from (BAI et al., 2009). (a) Contour with windows - the initial segmentation is given as input.. (a) Contour in the next frame. (c) Final result as calculated using the probability of each pixel belonging to the foreground.

Color model and color confidence: The color model M_c consists of two Gaussian Mixture Models, one for the background and one for the foreground. Only three Gaussians are used in each GMM. Having these two models, we can define the probability that the pixel belongs to the foreground as

$$p_c(x) = \frac{p_c(x \mid F)}{p_c(x \mid F) + p_c(x \mid B)}, \tag{4.19}$$

where $p_c(x \mid F)$ and $p_c(x \mid B)$ can be taken from the probability distributions. Gaussian Mixture Models are explained in Appendix C. The color confidence is used to describe how separable the colors from the foreground are from the background - that is, how reliable this classification is.

Shape model: The shape model consists of the segmentation, together with a confidence value that is related to the distance from the border.

Once initialized, these classifiers are propagated into the next frames using motion estimation. The color and shape classifiers are updated and the outputs of all classifiers are aggregated to generate a new segmentation.

4.5 Summary

This Chapter reviewed the existent literature in Computer Vision that is related to our proejct. The first part reviewed video registration, divided in two sections: feature detection and homography estimation. This organization mimics the first two steps of the pipeline used for this type of application. In the second part, we went over another important sub-area of Computer Vision that is used in our project: object tracking.

5 PANORAMIC VIDEOS FOR NON-LINEAR NAVIGATION

This chapter describes our solution for creating panoramic videos for non-linear navigation. It discusses the faced challenges and how we solved them. We will first discuss the interface and the features we want to provide. Then, we will focus on how to implement these features in the three different categories of videos we presented on Chapter 2. Note that while the implementation has specifities according to the type of video, our interface is conceptually general.

5.1 Interface

The key contribution of this work is the idea of a user-friendly interface for providing more effective navigation and interaction for e-learning videos. In this Section, we discuss the features it provides to the user.

5.1.1 Panoramic view

The first feature we want our videos to provide is a panoramic view of all the previously shown content of the lecture. For videos where there is a clear spatial relation between frames, we introduce the *extended classroom* metaphor: the idea that student sitting in the classroom would be able to see things not shown by the limited field of view of the current video frame. We go beyond showing the current content of the board/canvas, and provide immediate access to all content previously shown during the lecture (including erased and hidden content). Figure 5.1 shows two examples of panoramic images (i.e., extended frames) that illustrate this concept.

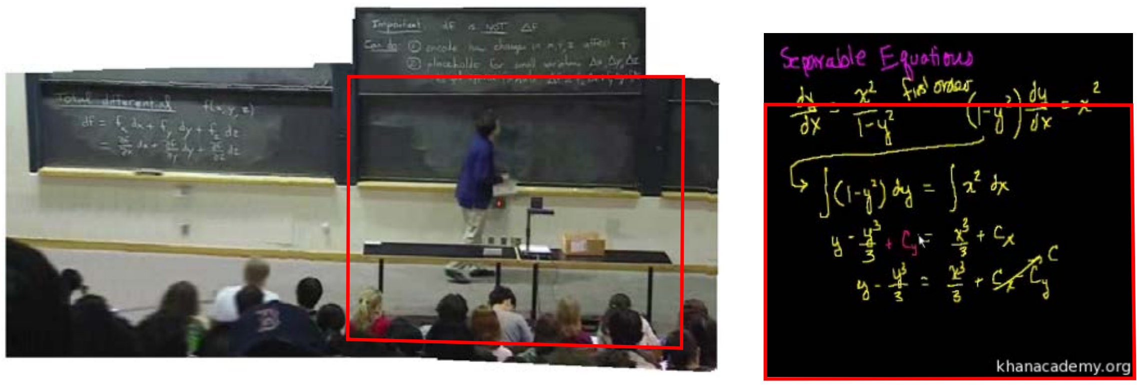


Figure 5.1: Extended video. Two examples of how the panoramic view of the lecture tries to simulate what the student would see if not restricted by the camera. The red rectangle shows the regular video frame at the same time.

5.1.2 Automatic generation of notes

The panoramic view shows the whole content of the lecture, making it a good summary to be exported as a set of lecture notes. Our interface also allows the user to annotate the video at any time, and export these annotations together with the panorama to a file. Figure 5.2 shows one annotated panoramic frame.

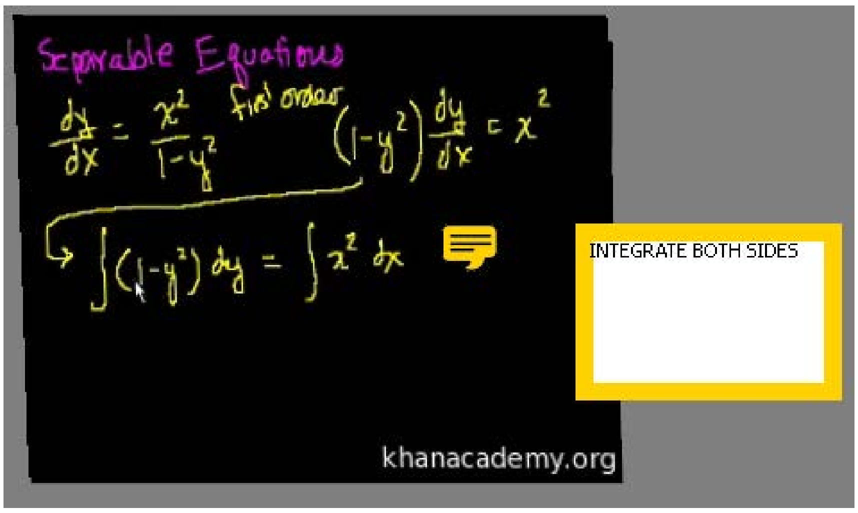


Figure 5.2: Automatic generation of notes. From our interface, one should be able to annotate the panoramic video and export the final notes to a file.

5.1.3 Hyperlinks

A key concept in our design is the notion of non-linear navigation through a pointing interface. We want the user to able to immediately access the instant when any topic first appeared by just pointing to it.

5.1.4 Extensions available on the fly

A very important feature in our interface, and also a very restricting one, is that everything should be calculated in real-time. It would be very hard to imagine that the user who wants to watch a video would sit and wait for hours. Also, many videos are watched by while they are being downloaded, which would not be possible if our technique involved offline calculations.

5.2 Creating Panoramic Videos

The first essential feature we provide in our videos is the extension to panoramic frames. As we discussed in Chapter 4, we need to register the frames to find a projective transformation from the individual video frames to a global coordinate system. We will discuss the approaches for canvas-based videos, slide-based videos, and recorded lectures separately. In all cases, for the feature detection, description and matching, we used a GPU implementation of SIFT (WU, 2007). Feature detection is performed separately to each frame of the video (Figure 5.3).

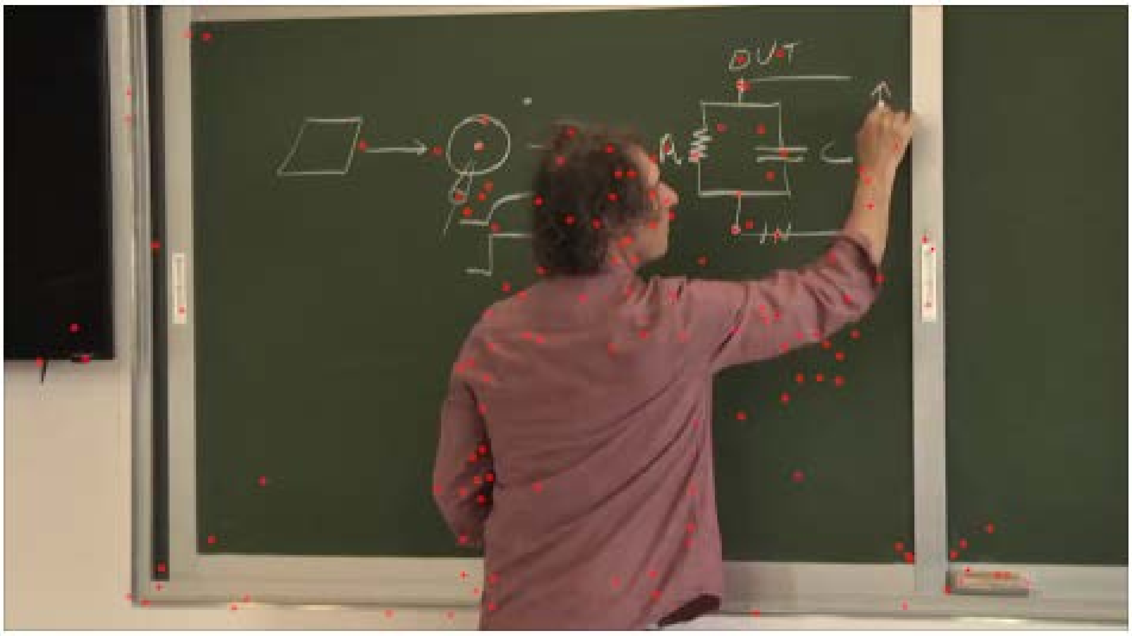


Figure 5.3: Sift points. Location of features detected in a frame, as calculated in our system.

5.2.1 Canvas-based videos

Canvas-based videos are the easiest to adapt to our technique. They have instrinsic spatial relations, high contrast and no occlusion problems.

Matching: In this type of video, there is a clear spatial relation from one frame to the next. Subsequent frames have significant overlap when transformed to a world coordinate system. Having significant overlapping regions is a pre-requisite for robust registration. In this case, we match the features in frame i to the ones in frame $i-1$.

Homography: Once the features are properly detected and matched, we need to estimate a homography between the sets of points. As we discussed in Section 4.3, we use RANSAC to ensure the estimation is robust to outliers. A high-level description of this algorithm follows:

- Randomly select four pairs of matched points from the frames i and $i-1$;
- Estimate a homography from these points, using SVD;
- Check which pairs are inliers to the homography, given a certain threshold;
- Repeat until you find enough inliers or reach the maximum number of iterations.

We estimate a homography for each group of four pairs using the Singular Value Decomposition implementation from CHOLMOD (CHEN et al., 2008). In terms of thresholding, we set our threshold t empirically between 0.01 and 0.0001. For the number of iterations, we use the approach described in Chapter 4, setting $\epsilon = 0.01$ initially and updating it at each iteration.

The estimated transformation maps the points from one video frame to the next. To find the global transformation, we multiply all homographies together. To create the panorama, we simply render the transformed frame over the existing ones.

5.2.2 Slide-based videos

Slide-based videos have, in comparison to canvas-based ones, the additional difficulty of not having explicit spatial information. Once we define an artificial position for every slide, the registration can be done very robustly. Again, we do not have the occlusion problems.

Matching: When dealing with slide-based videos, the approach discussed in Section 5.2.1 is not effective since slides do not overlap. However, since this type of lecture consists of a set of slides that can usually be downloaded with the video. In our system, we import these slides and place them side-by-side to create a big panoramic map of all content in the video. Once this is done, we match the features of each frame to this map.

Homography: In this case, we use a simpler approach to define the projective transformation of each frame. From our previous step, we already have the noise-free information of the position for the slide. Instead of estimating a homography, we calculate which slide has the most matches to the current frame. This strategy results in a fast and robust solution.

5.2.3 Recorded Lectures

Recorded lectures pose the biggest challenge to our technique among all types of e-learning video styles. The variability of this category is huge and the difficulties we encountered are far from solved problems in the current state-of-art.

The first part of the problem results from the fact that *the boards are often occluded by the instructor*. Also, the movement of the instructor can cause severe misregistrations if the board does not have many features. For this type of video, then, we added a step of pre-calculating a binary mask identifying which parts of the frames correspond to the instructor.

A second problem is that *we do not know in advance which parts of the frame belong to the board*. This is specially problematic when boards move. If many features move in one direction, this might be perceived by our algorithm as a movement of the camera between two frames, causing severe misregistration problems. If we detect and track boards, this problem can be partially solved, except for the cases when the board covers a majority of the frame, making it impossible to estimate the transformation from points outside it.

It is important to note that the goal of this thesis was not to propose new solutions in Computer Vision. With that in mind, we solved some of the problems - in some cases with a significant amount of user interaction - to the extent that was allowed by existent research.

Professor Tracking and SIFT Removal: To generate the binary mask of the instructor, we used the implementation of RotoBrush in Adobe AfterEffects (ADOBE AFTER EFFECTS, 2013). RotoBrush is used to detect the instructor's silhouettes (Figure 6.4a) from a set of user-provided scribbles. Given the silhouette, RotoBrush automatically computes a binary mask for segmenting the instructor (Figure 6.4b). The length of the videos made this a very difficult task, requiring user intervention from time to time. In our pipeline, the detected sift points that are inside this mask are removed from subsequent stages, *i.e.*, are not considered for the image registration and the creation of hyperlinks.

Board Segmentation: To segment the boards, we tried standard Hough transform (DUDA; HART, 1972), as well as the Kernel-based Hough transform (FERNANDES; OLIVEIRA, 2008). While the latter significantly out-performed the former, its results were still not robust enough to be applicable to our system. We decided to focus on more simple cases and leave the case of moving boards as a future path for exploration.

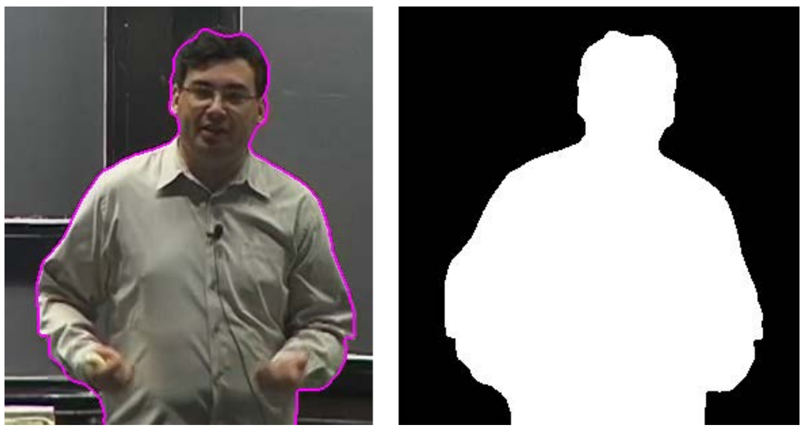


Figure 5.4: Professor tracking. (a) Contour of the professor, as tracked by RotoBrush. (b) Binary mask.

Recorded Lectures as Canvas-based Videos: Once we removed the problems of instructor occlusion and misregistration, we can see a recorded lecture simply as a canvas-based video. The registration and the generation of hyperlinks can all be done using the exactly same algorithm as the one described in the previous sections.

5.2.4 Erased Boards

In canvas-based videos and recorded lectures, content will sometimes be erased to give place to newer formulations. When this happens, we want the old content to be kept accessible somewhere. Our approach was to allow the user to move new content (soon to be rendered), thus extending the panoramic video frame. We do not detect automatically when a board is erased. We tested a few simple metrics to decide whether a new frame shows an erased board, such as the number of features, but these are not robust enough to handle all cases in a satisfactory way.

5.3 Creation of hyperlinks

The creation of hyperlinks is the most important extension we provide with our technique. Since we want non-linear navigation to be available in real-time, we cannot afford to process the whole video to find the hyperlinks. Instead, we rely on data we are already using for registration. When some text or an equation appears in the video, it usually contains one or more distinctive edges and, as such, it defines sift points. For each location in our reference coordinate system, we store the moment where a feature first appeared, and use it to navigate back when that content is selected.

More specifically, every time we calculate the transformation of a new frame into the global panorama we go through its sifts and test whether their respective locations already have a feature point. If not, the current frame is stored as the first appearance of the feature corresponding to that region. When a user clicks on the video, we find the nearest stored feature and navigate to the corresponding frame. To avoid excessive cluttering, we segment our clickable map in 10×10 regions with at most one feature each. The stored features for a video segment can be seen in Figure D.2.

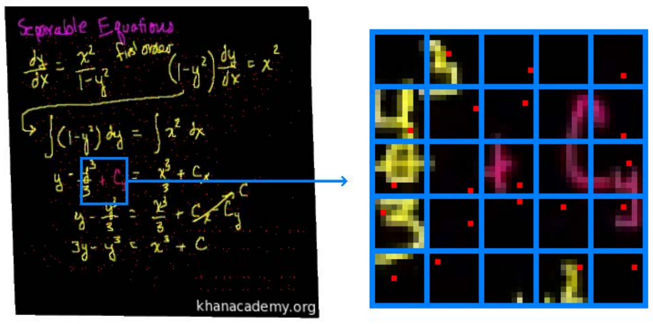


Figure 5.5: Creation of hyperlinks. The red dots represent feature points with associated hyperlinks (for non-linear navigation) stored at those particular locations. Note how there is at most one feature per cell.

In Figure D.2, we can see there are some red dots on regions that are not equations or text. This happens because, since each region keeps the first feature to appear inside it, the cursor might generate SIFT points that are stored as a hyperlink. This will generate unexpected behavior if the cursor appears much earlier in a region where later an important element will be drawn. Clicking that element will browse the video to the moment when the cursor was at that position. One strategy we tested to reduce this problem is to remove feature points in completely blank areas (when the cursor moves away, the area should be completely blank again). This approach greatly reduced the number of spurious hyperlinks in canvas-based videos. It is unfortunately not trivially generalized to more complex videos where the background does not have a constant color. A robust way to detect which feature points should be kept as hyperlinks is a future direction for exploration.

One issue that comes from non-linear navigation is that when we navigate back in time, for example, to the beginning of the video, we lose part of the content that was visible in the panorama (Figure 5.6(b)). While this is the expected behavior, it causes a problem when the user wants to return to the point where he was, or even to other explanations in between. To solve that, we give the user the option of seeing all the content shown so far by pressing a key. This "directory" is shown in gray shades to differ from the actual video frames (Figure 5.6(c)). By seeing the hyperlinks position, the user can navigate back to the desired moment in the video.

Another issue with allowing non-linear navigation is that we do not store the frames of our panoramic videos - instead, they are created by the renderization of subsequent frames on top of each other. If we jump in our navigation, this renderization will not create a panorama (Figure 5.7)(a). Also, when we navigate back in time, we do not want to keep future content visible (Figure 5.7)(b). To prevent that, every time our registration shows that a frame has moved significantly, we keep a panorama base for that moment saved in a buffer. The panoramic frame for any moment in the video can be restored using the panoramic base + the regular frame.

5.4 Automatic Generation of Notes

Our system automatically generates lecture notes from the panoramic video frames. For this, we simply export the panoramic view of the lecture together with the annotations the user made during the lecture. To handle PDF files robustly, we used the Poppler

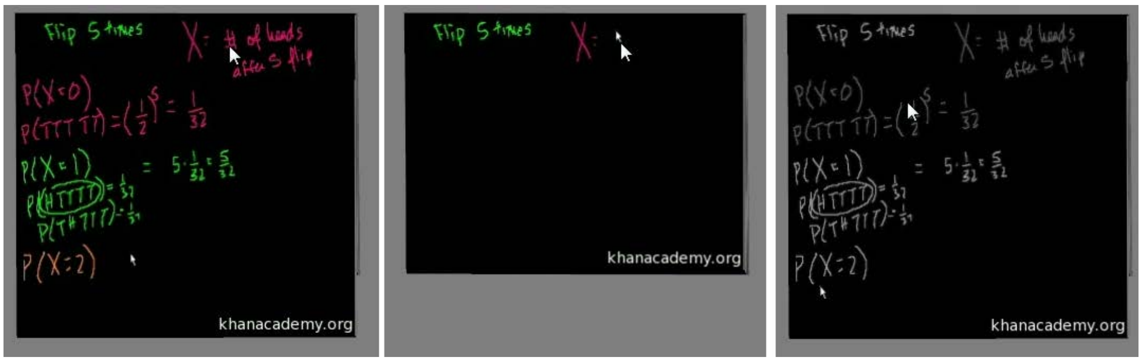


Figure 5.6: Forward navigation. Navigating back may cause the panorama to be erased. We allow the user to press a key to see the latest map available. (a) One panoramic frame. (b) After navigating to a point back in time, the user cannot see the content required for selection to move forward. (c) The "directory" shown in gray provides necessary information to allow the user to move to any of the contents presented so far.

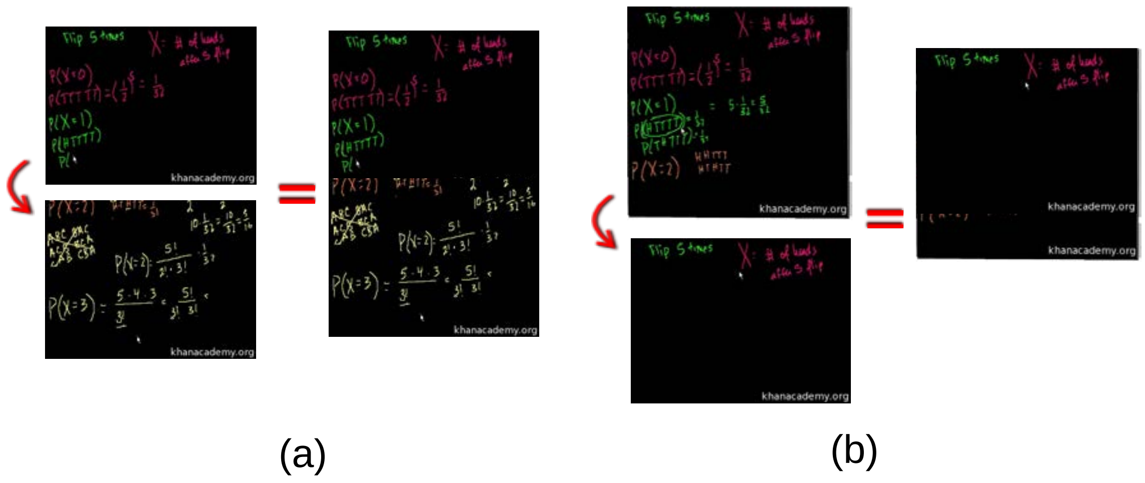


Figure 5.7: Jumps in navigation might destroy panorama. (a) If we jump from a frame in the beginning to one in the end (possible using the forward navigation discusse earlier), the panorama will not be formed. (b) Navigating back also causes problems.

library, that integrates nicely with the Qt Framework we are using for the interface.

5.5 Real-time Implementation

Ideally, our system should be able to handle videos in real time, without any pre-processing. We were able to achieve this for low-resolution videos (e.g., 320 $\times$ 240). To allow that, we divided the main workloads of our system into threads. The core computation is done using 3 other threads: one for the calculation of sifts, one for the estimation of transformations, and one for the creation of the buffered panoramic views that allow non-linear navigation.

Sift thread: The first thread computes sifts for every image in the video and matches them accordingly. This is the bottleneck of our computation and sometimes leads to less than real-time performance. The task is highly parallel: each image generates its own sifts and each match between a set of sifts from one image to another is also dependent only on that specific pair of images. In our setting, however, we performed every calculation sequentially due to the limited hardware. This thread is also responsible for removing sifts that are inside the instructor mask, in the case of recorded lectures.

Transformation thread: The second thread computes the transformation each frame has to undergo, based on the sifts calculated by the first thread, as well as the hyperlinks. It is necessary that the hyperlinks be calculated together with the transformation, since the sift thread only has information about the local position of each feature.

Map thread: The third core processing thread stores a panoramic view every time a frame moves away significantly from its predecessor. As we explained in Section 5.3, this is important to allow us to navigate non-linearly back and front without generating gaps in the panorama.

5.6 Summary

This Chapter described the features available our system. It also discussed the challenges involved in providing them to the three classes of e-learning videos handled by our system.

6 RESULTS

This Chapter illustrates the use of our technique applied to the three classes of e-learning videos described in Chapter 2. Since our technique involves processing and interacting with videos, it is hard to demonstrate the results using using only images. Thus, some short videos illustrating the use of our technique are available online at http://www.inf.ufrgs.br/~rgschneider/research/msc/videos/. We encourage the reader to watch them, as a way to get a better idea of the dynamic aspects of our method.

6.1 Panoramas

The first step in our pipeline is the generation of panoramic frames. As we described in Chapter 5, we detect the SIFT points for each video frame and match them, from one frame to the next. The homographies are estimated accordingly and the global transformation that each frame has to undergo is found by concatenating these local transformations.

Figure D.3(left) shows the panorama created for a lecture from Khan Academy. We can see how the view is significantly extended: the regular video resolution is 320×240, while the panoramic frame has resolution 320×644. Also, the panoramic frame has quality comparable to the original frames, *i.e.*, the content elements do not significantly lose sharpness. Figure D.3(right) illustrates the process of creating the panorama by rendering subsequent frames one over another.

Figure 6.2 shows a panorama of a conventional recorded lecture. Once again, the panorama significantly extends the field of view of the user: from regular video frames of resolution (480×268) we generated a panoramic frame of resolution (831×260). Also, even in the case of this low resolution video, we were able to generate a high quality panorama - note how the text is still perfectly recognizable and the result is visually satisfactory. In Figure D.4, we can see some of the frames that were used to generate the panorama and their respective final positions. Note how the final panorama does not have artifacts, even in the presence of significant zooming (Figure D.4, last frame).

6.2 Content-based Navigation

Hyperlinks are an inherently dynamic property of our interface. In order to demonstrate our results, we resorted to capturing some moments of the navigation: we show the panoramas and a set of hyperlinks, with the corresponding video frames they let the user navigate to.

Figure D.5 shows the results when using our technique with a canvas-based style video. The image on the left shows the reconstructed panorama, highlighting three hyper-

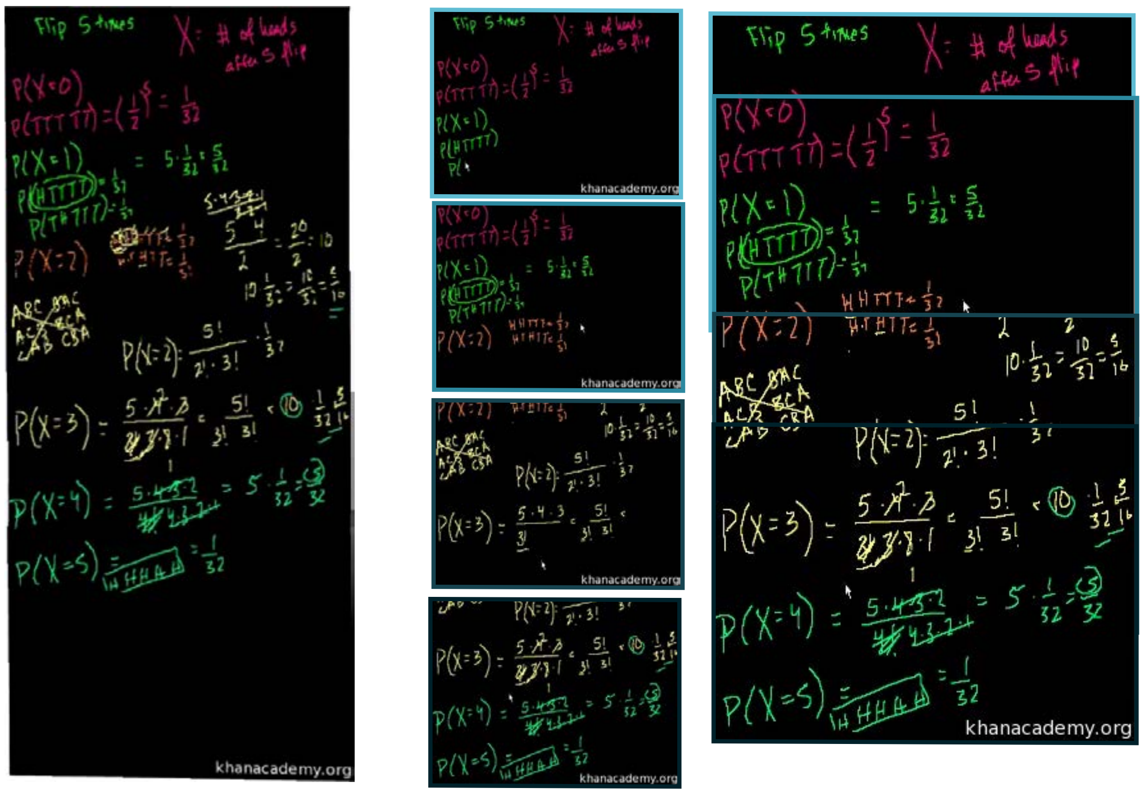


Figure 6.1: Panorama results for a lecture on probability from Khan Academy. (left) Final panorama result. (right) A subset of the frames that form the panorama.

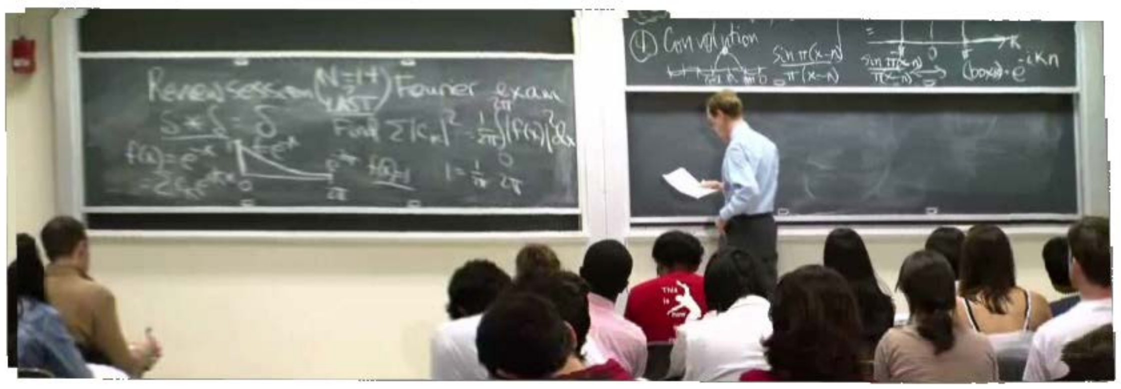


Figure 6.2: Panorama results for the Computational Science and Engineering Lecture on MIT OpenCourseWare.

links. The images on the right show the frames associated with the corresponding links. We also highlighted the position of the cursor in these target video frames, to show how our interface precisely navigates to the moment the content is being drawn on the screen.

Figure 6.5 shows another video from Khan Academy. In this case, the canvas was erased once, such that user intervention was needed to move the content and stop it from erasing existing equations. In this case, our interface simply extends the panorama to cover the new and old contents. Since hyperlinks are kept according to their position in the global coordinate system, the technique handles such cases naturally.

Figure D.6 shows the results for a slide-based style video. The panorama is shown

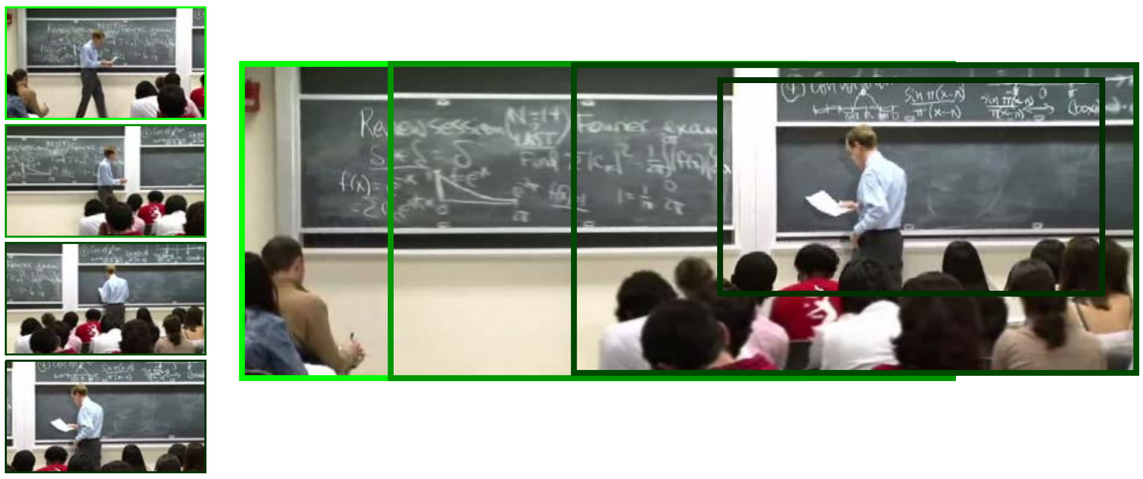


Figure 6.3: Some frames (left) used in conventional lecture panorama (right). Note how the system naturally handles camera zooming.

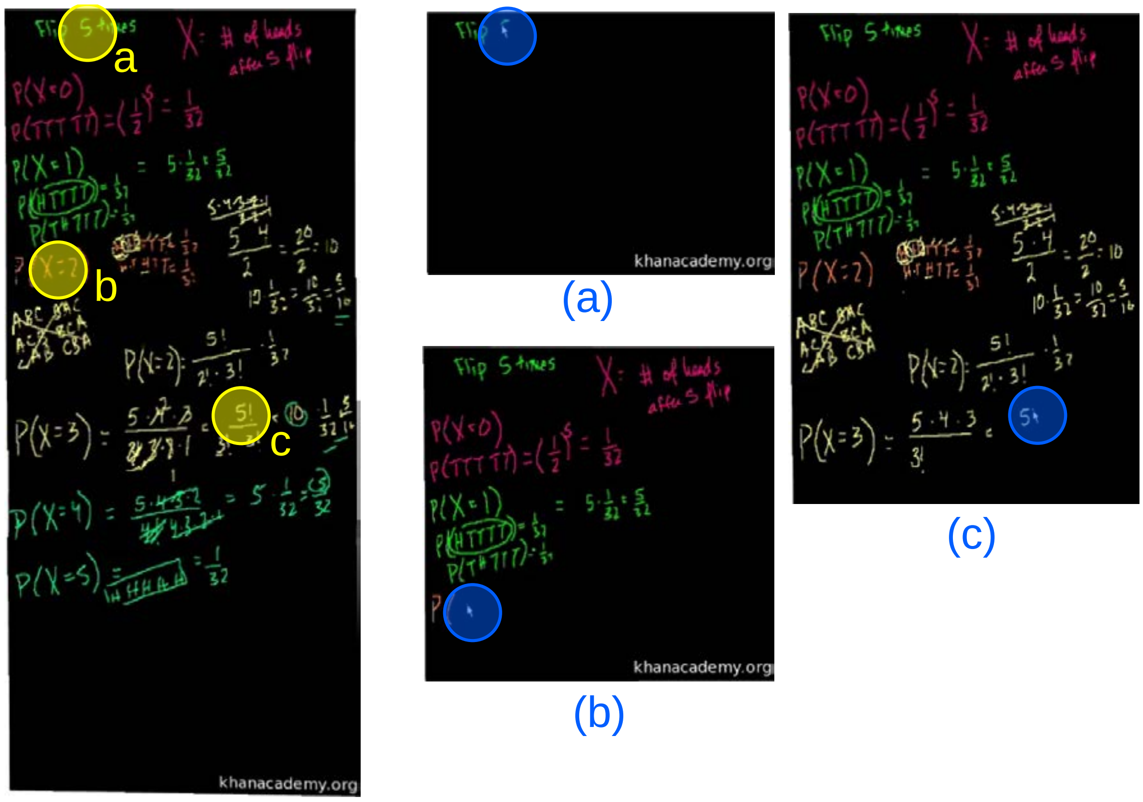


Figure 6.4: Results for a lecture on probability from Khan Academy. Panorama with some hyperlinks indicated, as well as the respective target video frames.

on the top. As explained in Chapter 5, this panorama is calculated by importing a set of slides from a file and putting them side-by-side. It is the same for the whole video, so we ommited the panoramic frames when showing the targets of the hyperlinks on the right. Note that in some cases, when all the text is showed directly in the first appearance of the slide, every hyperlink will point to the moment of that first appearance. In the result we show, the instructor writes on the slide, giving us the possibility to add those hyperlinks to the right time.

The final result on the content-based navigation is the conventional lecture recorded

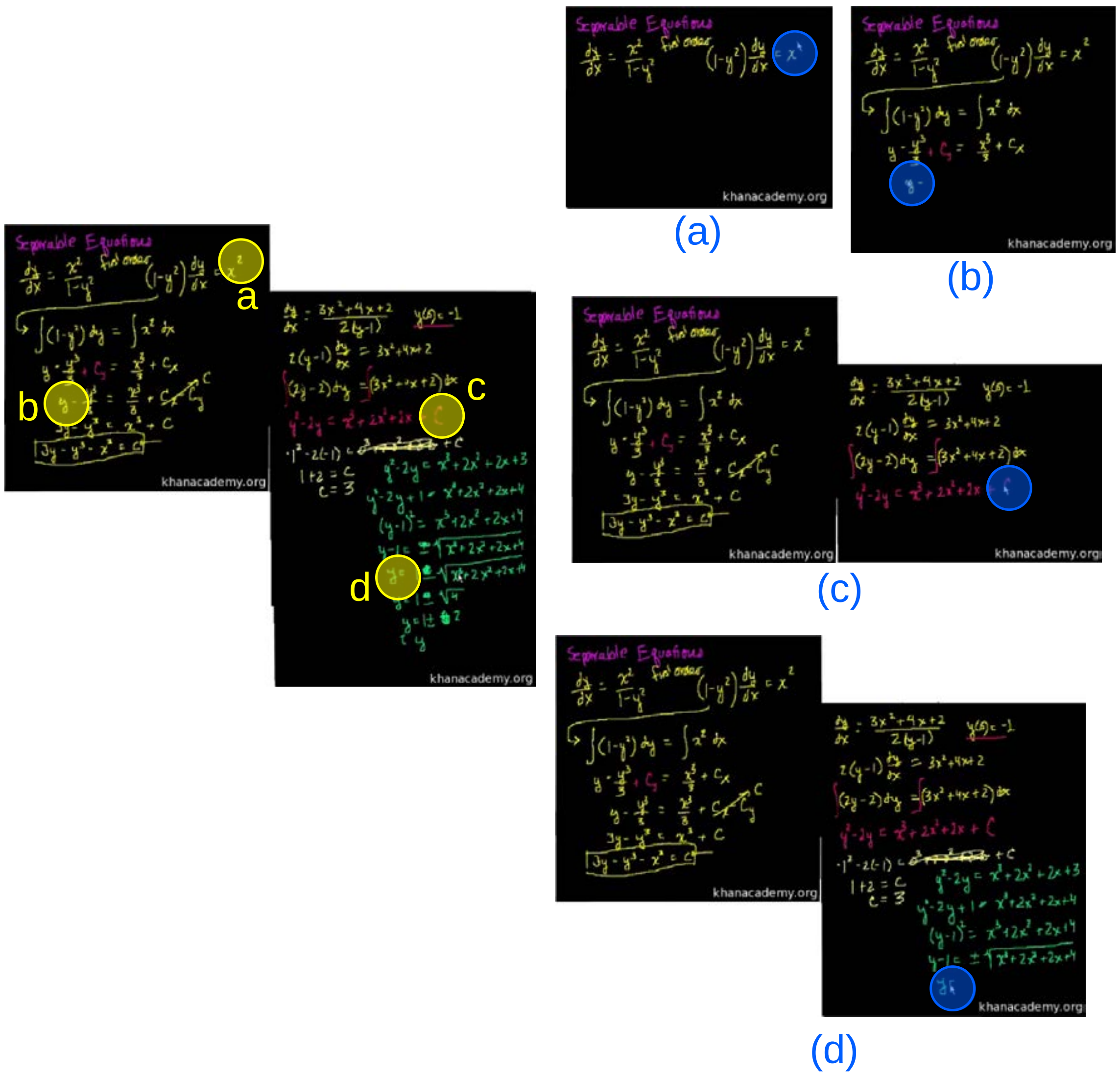


Figure 6.5: Results for a lecture on separable equations from Khan Academy. (left) Panorama with highlighted hyperlinks. (b) Target video frames.

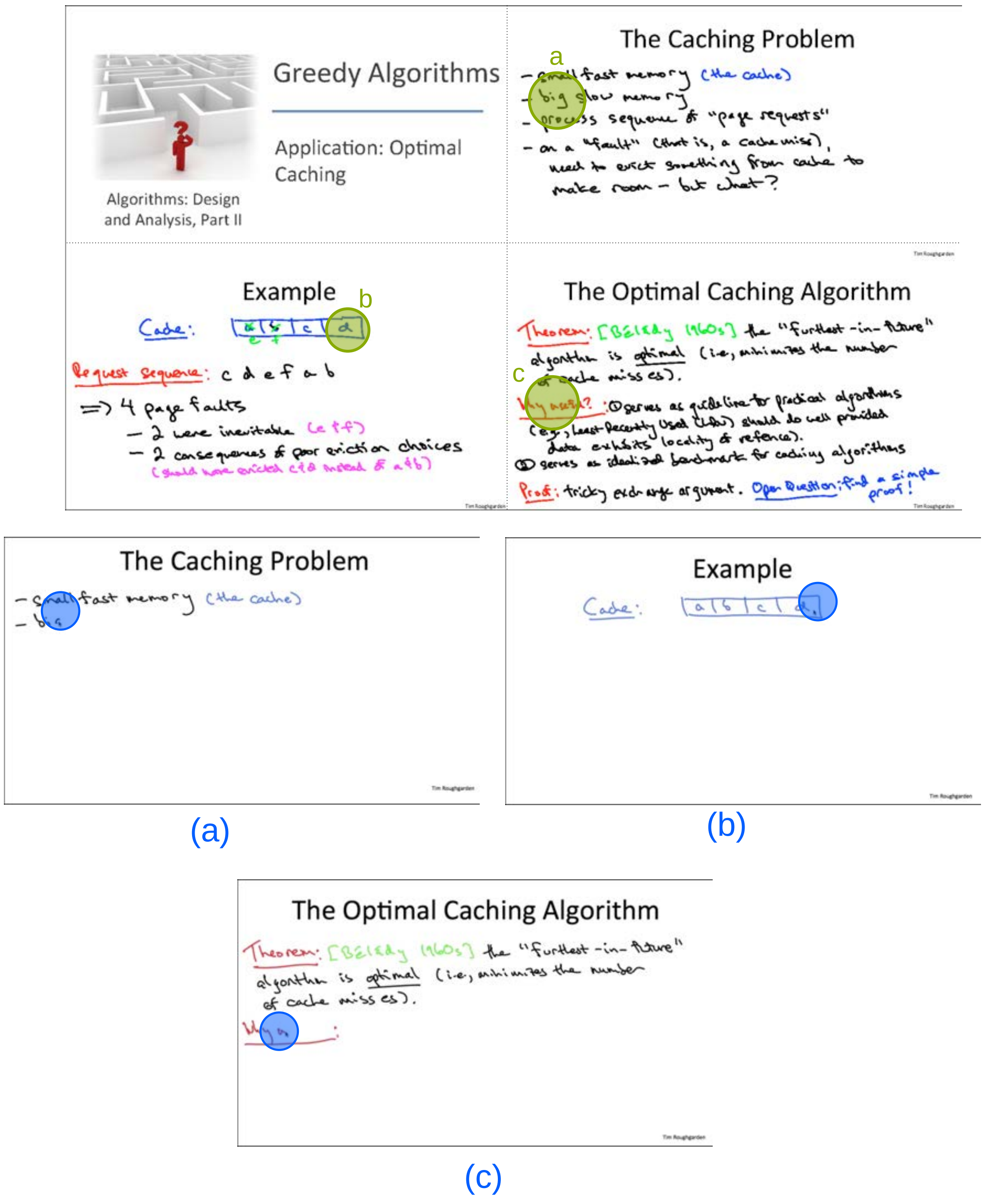
with a camera (Figure D.7). To demonstrate the hyperlinks, we used a lecture with high resolution. In this case there is no cursor, but we can see that in each of the target video frames the instructor is about to write the content that was selected.

To generate the conventional lecture panorama shown in Section 6.1, we used a short video segment (1min23s), to help us achieve good registration results. In the beginning of this video segment, most of the equations are already written on the board, so content-based navigation will not work. For one part that was written during the video segment, our algorithm found the right moment in the video (Figure 6.8).

6.3 Annotated Lectures Notes

Finally, our system also allows the automatic generation of lectures notes. The user can also annotate the video, and the annotations will be exported together with the final set of notes. We show the results for two videos to demonstrate this possibility.

In Figure D.8, we can see two moments of the video where the user is making notes. These were taken from our interface. Figure 6.10 shows the final set of exported notes, as



Greedy Algorithms

Application: Optimal Caching

Algorithms: Design and Analysis, Part II

The Caching Problem

- small fast memory (the cache)
- big slow memory
- process sequence of "page requests"
- on a "fault" (that is, a cache miss), need to evict something from cache to make room – but what?

Tim Roughgarden

Example

Cache: a b c d
e f

Request sequence: c d e f a b

⇒ 4 page faults
- 2 were inevitable (e & f)
- 2 consequences of poor eviction choices (should have evicted c & d instead of a & b)

Tim Roughgarden

The Optimal Caching Algorithm

Theorem: [Belady 1960s] the "furthest-in-future" algorithm is optimal (i.e., minimizes the number of cache misses).

Why useful?: ① serves as guideline for practical algorithms (e.g., Least Recently Used (LRU) should do well provided data exhibits locality of reference).
② serves as idealized benchmark for caching algorithms

Proof: tricky exchange argument. Open Question: find a simple proof!

Tim Roughgarden

Figure 6.6: Results for the Greedy Algorithms Lecture on Coursera. (top) Panorama. (bottom) Target video frames.

seen inside a PDF viewer.

In Figure 6.12, we can see the final annotated lecture notes. Again, to demonstrate that these annotations were made during the watching experience, we added two images of our interface at the moment when the notes were taken. We also show our notes inside a PDF Viewer to demonstrate that our notes are compatible with existing APIs.

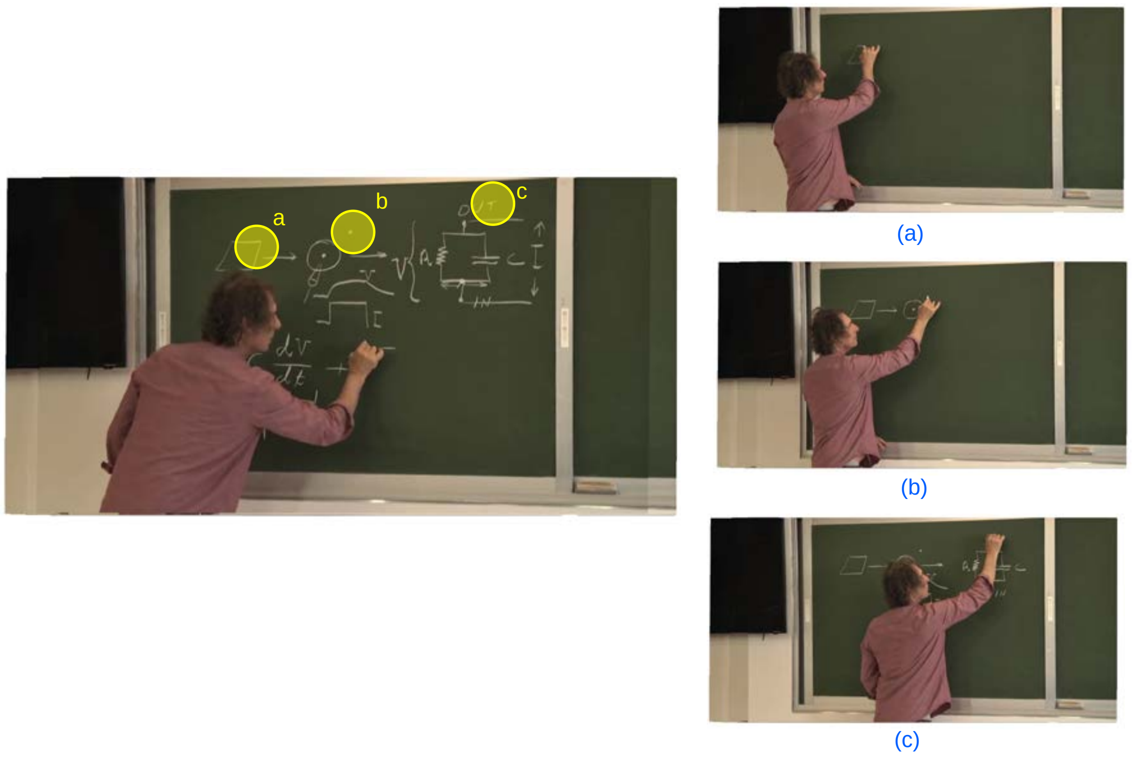


Figure 6.7: Results for the Synapses, Neurons and Brains Lecture on Coursera. (left) Panorama with highlighted hyperlinks. (b) Target video frames - note that the instructor is about to write the selected content.

6.4 Summary

This Chapter shows the results we achieved for the three classes of e-learning videos we discussed in Chapter 2. We demonstrate the three main features of our interface: panoramic frames, content-based non-linear navigation, and automatic generation of lecture notes.

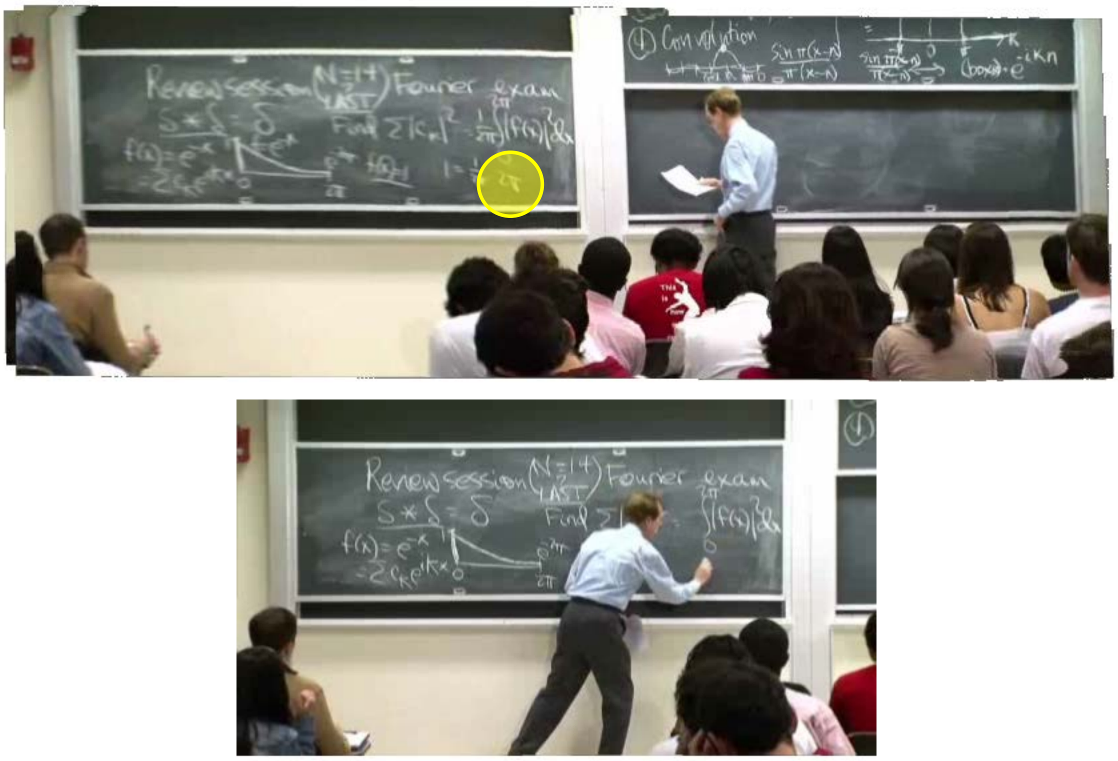


Figure 6.8: Results for the Computational Science and Engineering Lecture on MIT OpenCourseWare. (top) Panorama with one highlighted hyperlink - the video was too short to allow for more. (bottom) Target video frame.

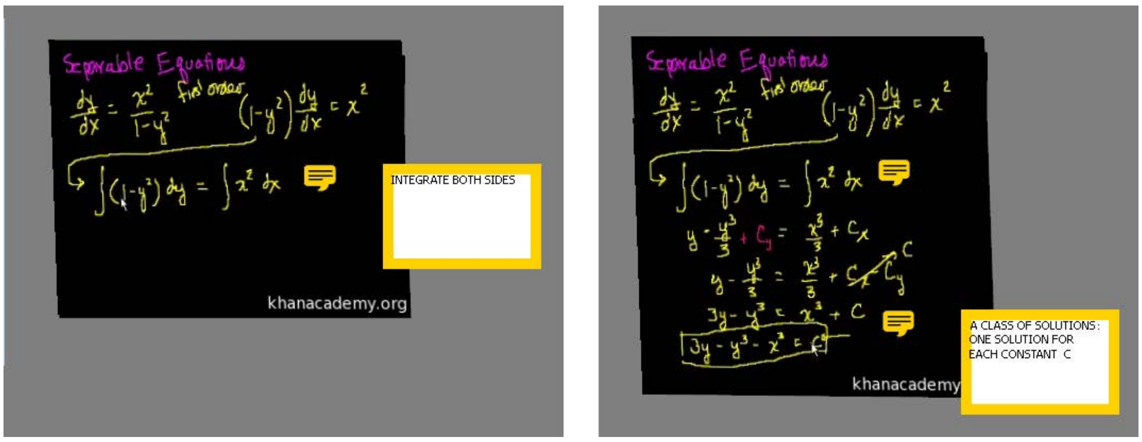
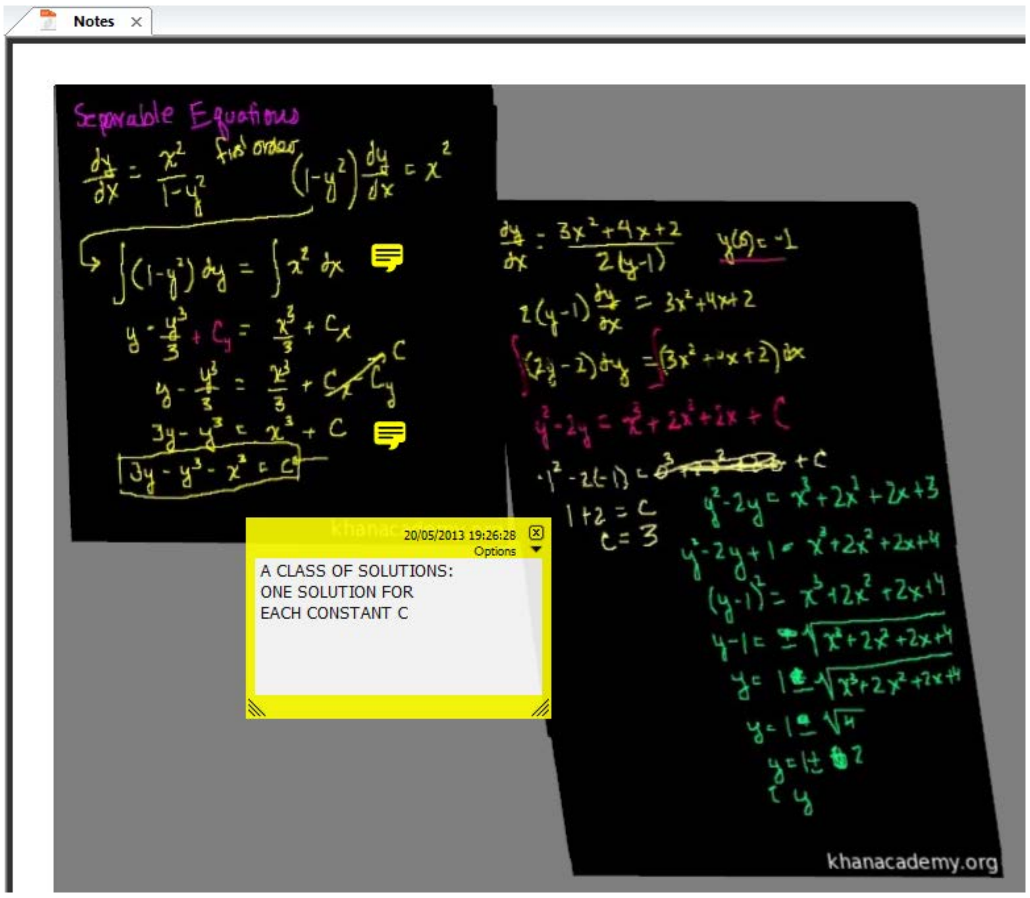
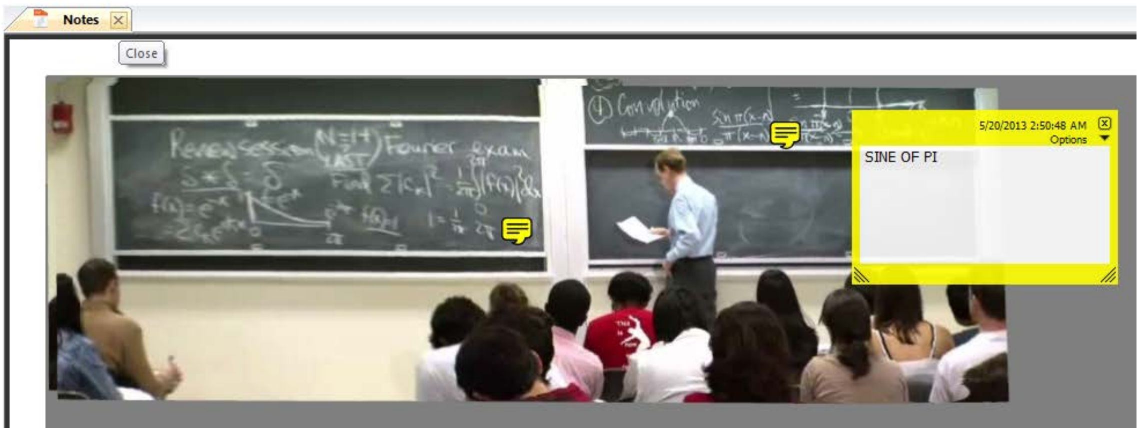


Figure 6.9: User can take notes while watching the video. (a) Video time: 3min07s .(b) Video time: 5min25s .



A CLASS OF SOLUTIONS:
ONE SOLUTION FOR
EACH CONSTANT C

Figure 6.10: Annotated panorama of a Khan Academy lecture with the respective notes, as seen inside a PDF Viewer.



SINE OF PI

Figure 6.11: Annotated panorama of a Khan Academy lecture with the respective notes, as seen inside a PDF Viewer.

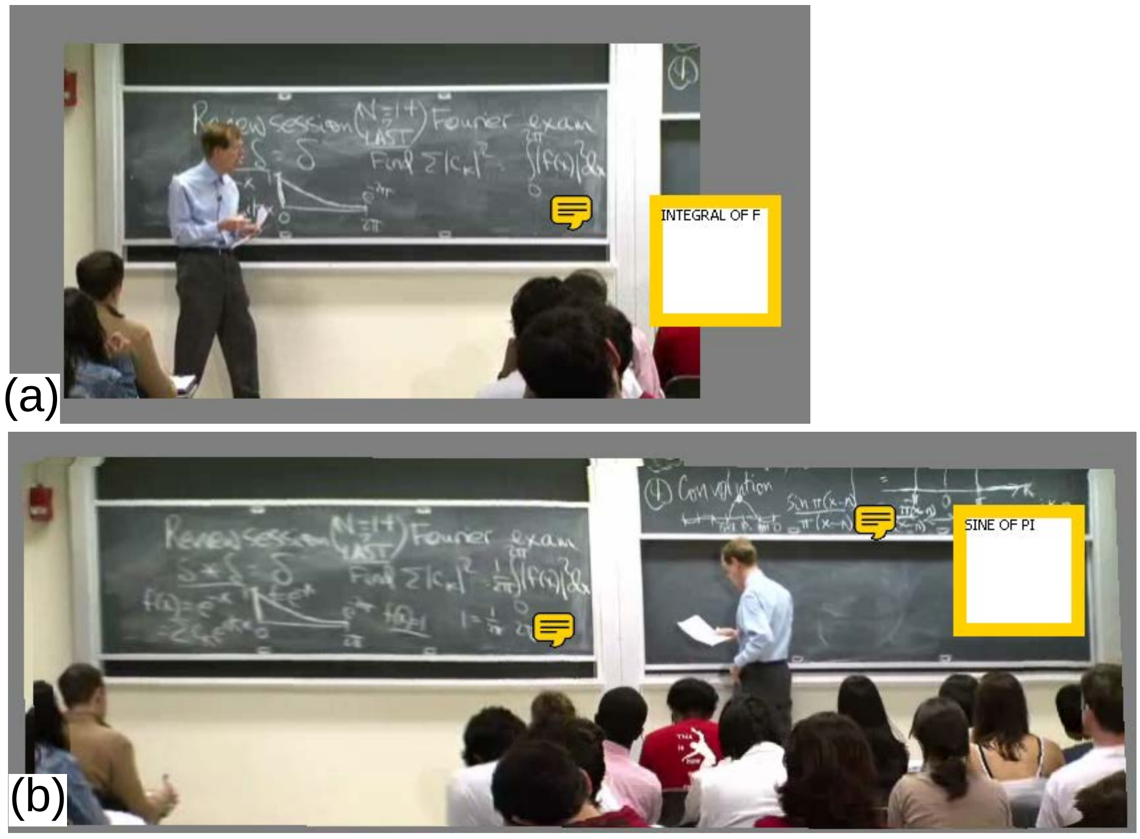


Figure 6.12: Annotated lecture notes from conventional lecture video. (a) Video time: 0min0s. (b) Video time: 1min23s.

7 CONCLUSIONS AND FUTURE WORK

This thesis presented a new interface for augmenting existing e-learning videos with panoramic frames and content-based navigation. Our technique improves current video players by allowing non-linear navigation through a simple pointing interface. We have demonstrated the effectiveness of the described concepts by applying them to three distinct classes of e-learning videos: canvas-based videos, slide-based videos, and recorded conventional lectures. For those, our system achieves real-time performance for low resolution videos (320 $\times$ 240). For higher resolution videos, some pre-processing is required for feature detection (using SIFT). However, since the most expensive parts of our processing pipeline are highly parallel, we believe that real-time performance might be soon achievable even for full HD resolution.

To create the panoramic frames, we detect relevant features in each frame, and use them to estimate a transformation of the frame into a global coordinate system. Once the frame is correctly mapped, simply rendering it over the others creates a panorama. The features are also stored to allow content-based navigation - the panorama is divided in 10 $\times$ 10 regions and the first appearance of a feature in each region is kept as the target frame of that hyperlink. Finally, we let the user export the final panorama as a set of annotated lecture notes. The techniques described in this thesis provide more efficient ways for exploring the benefits of e-learning videos. As such, they have the potential to impact education by providing more customizable learning experiences for millions of e-learners around the world.

Our system relies on Computer Vision techniques and, as such, inherits their limitations. First, the registration of videos does not give us satisfactory results in all cases. Having a robust registration step is essential to the creation of visually pleasing panoramas and to allow our hyperlinks to navigate to the correct moment in the video. Some works in Computer Vision (SZELISKI, 2010) suggest that generating SIFT points and matching them might not be the best approach for the registration of videos, which have very similar frames. Since this is a totally automatic step, we could allow more sophisticated methods, pre-calculating the final transformations and storing them for later use.

A second issue is that some sifts might not relevant to the actual content of the lecture. In canvas-based or slide-based video, one example would be the sifts generated by the cursor. In recorded lectures, the effect can come from shadows and even from noise in the video. One idea that might be used to overcome these issues is to track sifts for some time before adding them as a hyperlink. While this might significantly reduce the detection of irrelevant features, it may also cause some real hyperlinks to disappear if the corresponding features become occluded for some time.

A variety of other issues might come up when analyzing special cases of videos. For instance, when rendering the video in a panoramic frame, the lecturer might leave the regular video frame. If the instructor moves to an area where a part of the panorama is rendered (which of course consists only of the boards, and not of the instructor) it gives the impression that the professor disappeared. Also, the movement of boards was not treated in our implementation. Finding solutions for above issues and limitations provides interesting directions for future exploration.

REFERENCES

ADOBE After Effects. http://www.adobe.com/br/products/aftereffects.html. Accessed: 18/05/2013.

AGARWALA, A. et al. Panoramic video textures. In: ACM SIGGRAPH 2005 PAPERS, 2005, New York, NY, USA. **...** ACM, 2005. p.821–827. (SIGGRAPH '05).

ASSA, J.; CASPI, Y.; COHEN-OR, D. Action synopsis: pose selection and illustration. **ACM Trans. Graph.**, New York, NY, USA, v.24, n.3, p.667–676, July 2005.

BAE, S.; AGARWALA, A.; DURAND, F. Computational rephotography. **ACM Trans. Graph.**, New York, NY, USA, v.29, p.24:1–24:15, July 2010.

BAI, X. et al. Video SnapCut: robust video object cutout using localized classifiers. In: ACM SIGGRAPH 2009 PAPERS, 2009, New York, NY, USA. **...** ACM, 2009. p.70:1–70:11. (SIGGRAPH '09).

BARNES, C. et al. Video Tapestries with Continuous Temporal Zoom. **ACM Transactions on Graphics (Proc. SIGGRAPH)**, [S.l.], v.29, n.3, Aug. 2010.

BROWN, M.; LOWE, D. G. Automatic Panoramic Image Stitching using Invariant Features. **International Journal of Computer Vision**, [S.l.], v.74, n.1, p.59–73, 2007.

BROWN, M.; SZELISKI, R.; WINDER, S. Multi-Image Matching Using Multi-Scale Oriented Patches. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05) - VOLUME 1 - VOLUME 01, 2005., 2005, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2005. p.510–517. (CVPR '05).

BURNS, J. B.; HANSON, A.; RISEMAN, E. Extracting Straight Lines. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, [S.l.], v.PAMI-8, n.4, p.425–455, 1986.

CHEN, Y. et al. Algorithm 887: cholmod, supernodal sparse cholesky factorization and update/downdate. **ACM Trans. Math. Softw.**, New York, NY, USA, v.35, n.3, p.22:1–22:14, Oct. 2008.

CHEN, Y.; RUI, Y.; HUANG, T. JPDAF based HMM for real-time contour tracking. In: COMPUTER VISION AND PATTERN RECOGNITION, 2001. CVPR 2001. PROCEEDINGS OF THE 2001 IEEE COMPUTER SOCIETY CONFERENCE ON, 2001. **...** [S.l.: s.n.], 2001. v.1, p.I–543–I–550 vol.1.

CORREA, C. D.; MA, K.-L. Dynamic Video Narratives. **ACM Transactions on Graphics (Proc. SIGGRAPH)**, [S.l.], v.29, n.3, 2010.

CORSTEN, C. **DragonFly - Reviewing Lecture Recordings with Spatial Navigation**. 2009. Bachelor Thesis — RWTH Aachen University.

CORSTEN, C. DragonFly: spatial navigation for lecture videos. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 28., 2010, New York, NY, USA. **Proceedings. . .** ACM, 2010. p.4387–4392. (CHI EA '10).

COURSERA. `https://www.coursera.org/`. Accessed: 22/03/2013.

DAKSS, J. et al. **Hyperlinked Video**. [S.l.]: In SPIE Multimedia Systems and Applications, v. 3528, 1998.

DANIEL, G.; CHEN, M. Video Visualization. In: IEEE VISUALIZATION 2003 (VIS'03), 14., 2003, Washington, DC, USA. **Proceedings. . .** IEEE Computer Society, 2003. p.54–. (VIS '03).

DESOLNEUX, A.; MOISAN, L.; MOREL, J. michel. Computational Gestalts and Perception Thresholds. **Journal of Physiology - Paris**, [S.l.], v.97, p.2003, 2002.

DRAGICEVIC, P. et al. Video Browsing by Direct Manipulation. In: CHI '08: PROCEEDING OF THE TWENTY-SIXTH ANNUAL SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2008, New York, NY, USA. **. . .** ACM, 2008.

DUDA, R. O.; HART, P. E. Use of the Hough transformation to detect lines and curves in pictures. **Commun. ACM**, New York, NY, USA, v.15, p.11–15, January 1972.

EDX. `https://www.edx.org/`. Accessed: 22/03/2013.

FELS, S.; MASE, K. Interactive video cubism. In: ACM INTERNATION CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 1999., 1999, New York, NY, USA. **Proceedings. . .** ACM, 1999. p.78–82. (NPIVM '99).

FERNANDES, L. A. F.; OLIVEIRA, M. M. Real-time line detection through an improved Hough transform voting scheme. **Pattern Recogn.**, New York, NY, USA, v.41, p.299–314, January 2008.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, New York, NY, USA, v.24, p.381–395, June 1981.

FöRSTNER, W. A framework for low level feature extraction. In: EUROPEAN CONFERENCE ON COMPUTER VISION (VOL. II), 1994, Secaucus, NJ, USA. **Proceedings. . .** Springer-Verlag New York: Inc., 1994. p.383–394. (ECCV '94).

GIOI, R. von et al. LSD: a fast line segment detector with a false detection control. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, [S.l.], v.32, n.4, p.722–732, 2010.

GOLDMAN, D. B. et al. Schematic storyboarding for video visualization and editing. In: ACM SIGGRAPH 2006 PAPERS, 2006, New York, NY, USA. **...** ACM, 2006. p.862–871. (SIGGRAPH '06).

GOLDMAN, D. B. et al. Video object annotation, navigation, and composition. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 21., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.3–12. (UIST '08).

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: IN PROC. OF FOURTH ALVEY VISION CONFERENCE, 1988. **...** [S.l.: s.n.], 1988. p.147–151.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2 ed. [S.l.]: Cambridge University Press, ISBN: 0521540518, 2004.

HERMANS, C. et al. Augmented Panoramic Video. **Comput. Graph. Forum**, [S.l.], v.27, n.2, p.281–290, 2008.

HESS, R.; FERN, A. Improved Video Registration using Non-Distinctive Local Image Features. In: COMPUTER VISION AND PATTERN RECOGNITION, 2007. CVPR '07. IEEE CONFERENCE ON, 2007. **...** [S.l.: s.n.], 2007. p.1–8.

HOLMAN, D. et al. Fly: an organic presentation tool. In: CHI '06: CHI '06 EXTENDED ABSTRACTS ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2006, New York, NY, USA. **...** ACM, 2006. p.863–868.

HORN, B. K.; SCHUNCK, B. G. **Determining Optical Flow**. Cambridge, MA, USA: [s.n.], 1980.

HOUGH, P. **Method and means for recognizing complex patterns**. 1962.

IMPA - Instituto Nacional de Matematica Pura e Aplicada. `http://video.impa.br/`. Accessed: 27/03/2013.

ISARD, M.; BLAKE, A. CONDENSATION - Conditional Density Propagation forVisual Tracking. **Int. J. Comput. Vision**, Hingham, MA, USA, v.29, n.1, p.5–28, Aug. 1998.

ITUNESU. `http://www.apple.com/education/itunes-u/`. Accessed: 07/05/2013.

KANG, H.-W. et al. Space-Time Video Montage. In: COMPUTER VISION AND PATTERN RECOGNITION, 2006 IEEE COMPUTER SOCIETY CONFERENCE ON, 2006. **...** [S.l.: s.n.], 2006. v.2, p.1331–1338.

KARRER, T. et al. DRAGON: a direct manipulation interface for frame-accurate in-scene video navigation. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.247–250. (CHI '08).

KHAN Academy. `https://www.khanacademy.org/`. Accessed: 22/03/2013.

KIMBER, D. et al. Trailblazing: video playback control by direct object manipulation. In: MULTIMEDIA AND EXPO, 2007 IEEE INTERNATIONAL CONFERENCE ON, July. **...** [S.l.: s.n.], July. p.1015–1018.

KWATRA, V. et al. Graphcut Textures: image and video synthesis using graph cuts. **ACM Transactions on Graphics, SIGGRAPH 2003**, [S.l.], v.22, n.3, p.277–286, July 2003.

LOWE, D. G. Object Recognition from Local Scale-Invariant Features. In: INTERNATIONAL CONFERENCE ON COMPUTER VISION-VOLUME 2 - VOLUME 2, 1999, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 1999. p.1150–. (ICCV '99).

MA, Y.-F.; ZHANG, H.-J. A model of motion attention for video skimming. In: IMAGE PROCESSING. 2002. PROCEEDINGS. 2002 INTERNATIONAL CONFERENCE ON, 2002. **...** [S.l.: s.n.], 2002. v.1, p.I–129–I–132 vol.1.

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.27, n.10, p.1615–1630, 2005.

MIT OCW. `http://ocw.mit.edu/index.htm`. Accessed: 22/03/2013.

MUJA, M.; LOWE, D. G. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: VISAPP (1), 2009. **...** [S.l.: s.n.], 2009. p.331–340.

NATIONAL Programme on Technology Enhanced Learning. `http://nptel.iitm.ac.in/`. Accessed: 27/03/2013.

NGUYEN, C.; NIU, Y.; LIU, F. Video summagator: an interface for video summarization and navigation. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p.647–650. (CHI '12).

POWELL, S.; YUAN, L. MOOCs and Open Education: implications for higher education. , [S.l.], 2013.

REYNOLDS, D. A. Gaussian Mixture Models. In: **Encyclopedia of Biometrics**. [S.l.: s.n.], 2009. p.659–663.

ROSE, K. K. Student Perceptions of the Use of Instructor-Made Videos in Online and Face-to-Face Classes. **Learning**, [S.l.], v.5, n.3, p.487–495, 2009.

SAND, P.; TELLER, S. Video matching. In: ACM SIGGRAPH 2004 PAPERS, 2004, New York, NY, USA. **...** ACM, 2004. p.592–599. (SIGGRAPH '04).

SATOU, T. et al. CyberCoaster: polygonal line shaped slider interface to spatio-temporal media. In: ACM INTERNATIONAL CONFERENCE ON MULTIMEDIA (PART 2), 1999, New York, NY, USA. **Proceedings...** ACM, 1999. p.202–. (MULTIMEDIA '99).

SHAH, R.; NARAYANAN, P. Interactive Video Manipulation using Object Trajectories and Scene Backgrounds. **Circuits and Systems for Video Technology, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–1, 2013.

SHAH, R.; NARAYANAN, P. J. Trajectory based video object manipulation. In: MULTIMEDIA AND EXPO (ICME), 2011 IEEE INTERNATIONAL CONFERENCE ON, 2011. **...** [S.l.: s.n.], 2011. p.1–4.

SHI, J.; TOMASI, C. Good features to track. In: COMPUTER VISION AND PATTERN RECOGNITION, 1994. PROCEEDINGS CVPR ’94., 1994 IEEE COMPUTER SOCIETY CONFERENCE ON, 1994. **...** [S.l.: s.n.], 1994. p.593 –600.

SMITH, J. M.; STOTTS, D.; KUM, S.-U. An orthogonal taxonomy for hyperlink anchor generation in video streams using OvalTine. In: ACM ON HYPERTEXT AND HYPERMEDIA, 2000, New York, NY, USA. **Proceedings...** ACM, 2000. p.11–18. (HYPERTEXT ’00).

SMITH, M.; KANADE, T. **Video Skimming for Quick Browsing based on Audio and Image Characterization**. Pittsburgh, PA: Computer Science Department, 1995. (CMU-CS-95-186).

SMITH, M.; KANADE, T. Video Skimming and Characterization through the Combination of Image and Language Understanding. In: IEEE INTERNATIONAL WORKSHOP ON CONTENT-BASED ACCESS OF IMAGE AND VIDEO DATABASES, 1998., 1998. **Proceedings...** [S.l.: s.n.], 1998. p.61 – 70.

SNAVELY, N.; SEITZ, S. M.; SZELISKI, R. Photo tourism: exploring photo collections in 3d. In: SIGGRAPH CONFERENCE PROCEEDINGS, 2006, New York, NY, USA. **...** ACM Press, 2006. p.835–846.

STRANG, G. **Linear algebra and its applications**. [S.l.]: Thomson, Brooks/Cole, 2006.

SZELISKI, R. Computer Vision : algorithms and applications. **Computer**, [S.l.], v.5, p.832, 2010.

TEODOSIO, L.; BENDER, W. Salient stills. **ACM Trans. Multimedia Comput. Commun. Appl.**, New York, NY, USA, v.1, n.1, p.16–36, Feb. 2005.

TRIGGS, B. Detecting Keypoints with Stable Position, Orientation, and Scale under Illumination Changes. In: ECCV (4)’04, 2004. **...** [S.l.: s.n.], 2004. p.100–113.

TRUONG, B. T.; VENKATESH, S. Video abstraction: a systematic review and classification. **ACM Trans. Multimedia Comput. Commun. Appl.**, New York, NY, USA, v.3, n.1, Feb. 2007.

UNDERSTAND Mathematics. `http://www.youtube.com/user/DrChrisTisdell`. Accessed: 26/03/2013.

UYTTENDAELE, M.; EDEN, A.; SZELISKI, R. Eliminating ghosting and exposure artifacts in image mosaics. **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001**, [S.l.], v.2, p.II–509–II–516, 2001.

WALTHER-FRANKS, B. et al. Dragimation: direct manipulation keyframe timing for performance-based animation. In: GRAPHICS INTERFACE 2012, 2012, Toronto, Ont., Canada, Canada. **Proceedings...** Canadian Information Processing Society, 2012. p.101–108. (GI ’12).

WANG, J.; COHEN, M. F. **Image and Video Matting**. Hanover, MA, USA: Now Publishers Inc., 2008.

WU, C. **SiftGPU**: a GPU implementation of scale invariant feature transform (SIFT). 2007.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: a survey. **ACM Comput. Surv.**, New York, NY, USA, v.38, n.4, Dec. 2006.

ZHANG, D. et al. Instructional video in e-learning: assessing the impact of interactive video on learning effectiveness. **Information & Management**, [S.l.], v.43, n.1, p.15 – 27, 2006.

APPENDIX A IMAGE COMPOSITION

When matching real images, it is common that two pictures have different exposition or objects in different positions. To create a panorama without artifacts, we need to choose which pixels will contribute to the final image, and how much.

The simplest possible approach is to take the average of all pixelspixels. Unfortunately, it does not perform very well. A better filter already is the *median filter*. It works well for removing objects that do not stay in any position for a long time.

A third possibility is to weight pixel contribution according to the distance from the center of the image. Pixels at the center are less likely to overlap and to have radial distortion. This process is usually called *feathering*. When the image has some cut-out regions, this can be done by using a distance map.

A.1 Optimal Seam Selection

In many situations, the overlapping regions of the images will be different from each other in more than just the tone of the colors, even if they are correctly aligned. The main reason is that the scenes we are interested in stitching are, mostly, not completely static - some of the elements might be in different positions when we use different views. In this case, we will assume we are not interested in which position the objects will be on the final panorama, as long as the final result is visually satisfactory.

A relevant fact is that the most noticeable artifacts come from moving objects. Uyttendaele et al. (UYTTENDAELE; EDEN; SZELISKI, 2001) propose a method for solving this problem, where the first step is to detect movement by finding which parts of the scene differ in different views. These regions are called regions of difference (RODs). The main idea is that only one image will be used to fill a ROD, avoiding seams. To select which image to use, they create a graph where intersecting RODs are connected by edges - and remove the vertex cover of the graph until only disconnected nodes remain.

A different approach to generating seamless merging of images is Graphcut Textures (KWATRA et al., 2003). Their technique starts from two overlapping images and selects an optimal seam to divide them. It works by creating a graph where each pixel is a node connected to its 8-neighborhood, as shown in Figure A.1. A simple approach for the cost of each edge is the difference of the color of the pixels in image. Let $A(s)$ and $B(s)$ be the colors of pixel s in the images A and B, respectively, the cost function for an edge between two neighboring pixel s and t would be given by

$$M(s,t,A,B) = \|A(s) - B(s)\| + \|A(t) - B(t)\| \tag{A.1}$$

Then, some pixels are denoted to be necessarily part of one image or the other - the non-overlapping regions, if any, are good candidates. Finally, a min-cut algorithm is used to divide the graph into two parts: the pixels/nodes belonging to the first image and the pixels/nodes beloging to the second one.

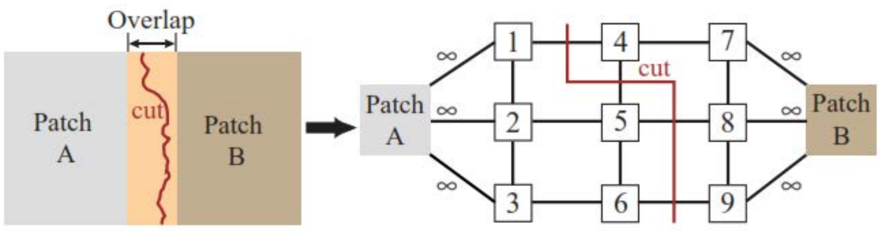


Figure A.1: Graph-cut textures. A graph is used to represent the image and a min-cut algorithm to find the optimal seam. Image taken from (KWATRA et al., 2003).

A more sophisticated cost function for the edges is to take into account how homogeneous the region being cut is. The reason for that is that a discontinuity in the merging of the two images is less likely to be noticed if it is along an image edge. To add this factor to our function, we make it inversely proportional to the gradient of the image in the region.

$$M'(s,t,A,B) = \frac{\|A(s) - B(s)\| + \|A(t) - B(t)\|}{\|G_A^d(s)\| + \|G_A^d(t)\| + \|G_B^d(s)\| + \|G_B^d(t)\|} \tag{A.2}$$

where $G_A^d(s)$ is the gradient of the image A in the pixel p along direction d. The direction used is the direction of the edge.

APPENDIX B LINE DETECTION

Line segments are very interesting in describing the geometric content of an image. Human-made objects are often composed of planar surfaces which that a description using lines can be a expressive and very economic representation.

B.1 Hough Transform

The Hough transform (HOUGH, 1962) is a voting scheme to detect straight lines in images. The original idea is that every edge pixel on the image votes for all potential lines passing through it (Figure B.1). Here, we understand an edge pixel as one whose image gradient is above a certain threshold. The votes are stored and the most voted lines are then picked as the probable real lines of the image. This maps the problem of finding the most prominent lines to the much simpler problem of finding peaks of a function. Once the vote peaks are selected, we can perform a least squares fitting of the line to the points, to achieve a more robust result.

Lines are represented as a slope θ and a distance r from the origin, so the domain of voting is two dimensional - as can be seen in Figure B.1. The set of all lines that go through a point can be represented by a sinusoid in the (r, θ)-plane - often, however, these parameters will be quantized into bins. This representation is not part of the original patent (HOUGH, 1962), and was introduced later by Duda and Hart (DUDA; HART, 1972), resulting in great memory savings.

Alternative representations of the domain are the use of the 2d normal $\hat{n}$ instead of the slope, resulting a 3D voting space $m = (\hat{n}_x, \hat{n}_y, r)$. A reasonable way to represent this voting domain is to use a cubemap. In fact, if we restrict r to be positive, only 3 faces of the cube will be used.

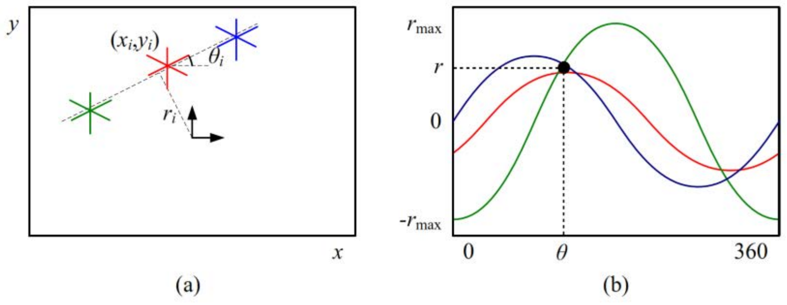


Figure B.1: Hough transform voting scheme. (a) Three points voting for all lines they could belong to - the most voted line is the one they are all in. (b) Intersection in (r, θ)-space represents real line.

Kernel-based Hough Transform. A more recent alternative to the original Hough transform is to use a kernel-based scheme to cluster groups of approximately collinear edge pixels and vote for each such a group at once using a bivariate Gaussian distribution based on the variances of the points. The technique provides a much faster voting scheme and a cleaner accumulator.

B.2 LSD

Recently, Gioi et al. (GIOI et al., 2010) introduced a new algorithm for line segment detection that provides significantly better results than Hough Transform. Their algorithm is divided into three main steps: partition the image into line-support regions, find the best line segment for each region and validate the line segment. According to the authors, their technique was based on two other works, Burns et al. (BURNS; HANSON; RISEMAN, 1986), that first suggested this pipeline, and Desolneux et al. (DESOLNEUX; MOISAN; MOREL, 2002), whose *a contrario* approach is used in the third step.

Line-support regions. When the algorithm starts, we have in our region only one pixel as a seed and one angle - the line-angle, that is orthogonal to the gradient at that pixel. At each step, the adjacent pixels - in a 8-neighborhood - are tested: if their orientation coincides to the region angle (the average of line-angles), up to a certain threshold, they are added to the region, and the region angle is updated. We always choose the pixels with greater gradient magnitude for the seeds, since they have higher probability of lying in a straight line.

Finding best line segment. Once we have a region, we define a straight line segment that best describes this region. In LSD, the line segment - with length, width and angle - is represented by a rectangle. To find the center of the rectangle, the authors propose using the center of mass, where the mass of each pixel is given by the magnitude of its gradient. The orientation is chosen using the first axis of inertia, and the width and length are set such that the rectangle covers the whole region.

Validating line segment. To validate a line segment, we try to model the background and statistically determine that the segment does not belong to the same probability distribution. The authors propose that a good model for the background is to assume that gradient angles are independent and uniformly distributed.

Inside each rectangular region detected by step two, we count how many pixels are aligned to the region-angle - up to a certain threshold. We do a hypothesis test to see whether the line segment is part of the background.

APPENDIX C GAUSSIAN MIXTURE MODELS

Gaussian Mixture Models, or GMMs, is probabilistic model for representing the presence of subpopulations within a dataset. We will discuss how we apply this model to the representation of images.

C.1 Sparse Modeling

For the task of removing noise from an image, we usually want to minimize the error from the recovered image to the original one. Suppose y is the measured image and x is the one we are trying to find, we want to minimize $|x - y|_2^2$. Now, unless we have some additional constraint, the solution to this equation is simply setting $x = y$. A very useful ideia is to define a model that we believe is true for good, noise free images, and use that as a constraint. This model is usually called a **prior**, and we will represent it by $G(x)$. Note that our equation is now given by $error(x) = |x - y|_2^2 + G(x)$.

Many different priors have been suggested over the years. To give an illustration, one may say that a natural image without noise should be smooth. One way to encode that as a prior is to minimize the magnitude of the Laplacians, $G(x) = |\mathbf{L}x|_2^2$.

A different idea, that we are mostly interested in, is to use a dictionary to represent the prior of the image. For example, suppose we want to represent a patch p, which is 8x8. We can do that by creating a dictionary D of 8x8 patches and representing p as a linear combination of these patches - called **atoms**. We represent that by setting $y = D\alpha$, where D is out dictionary and α has the coefficients of each patch. See Figure C.1. This model generalizes, for example, the DCT and the Fourier transform. We can define each of the atoms in the dictionary D to be one vector of the, for example, cosine basis.

Now, we want our representation to be sparse, that is, we want the patch to be represented by a small number of atoms from the dictionary. The assumption is that if we use only a few atoms, we will not be able to fit the noise well, resulting in a good image. To measure the sparsity of our representation, we can use the L_0 norm of the vector, which is equivalent to the number of non-zero elements. Our error equation will be as follows.

$$error(x) = |D\alpha - y|_2^2 \, s.t. \, |\alpha|_0^0 < L \tag{C.1}$$

One important observation, when treating images, is that having a dictionary for the whole image would be hard. Instead, we can divide the image into patches and represent the final color of the pixel by the average of all the patches that touch it. The final equation would be as follows, where R_{ij} is a matrix that extracts the patch around the pixel at row i and column j and μ is a normalization factor.

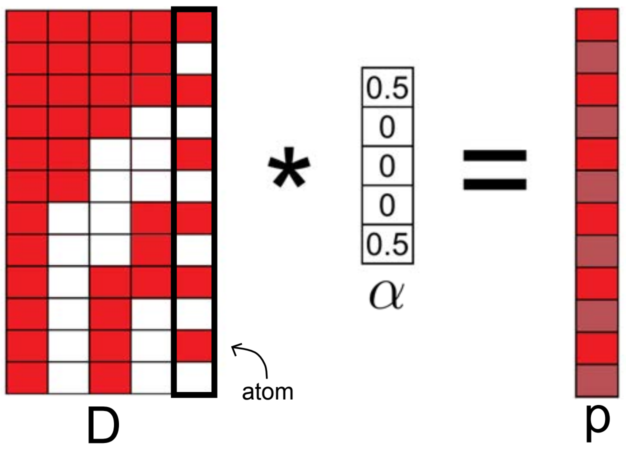


Figure C.1: Sparse modeling. The image can be represented by a linear combination of atoms in the dictionary - in the case of sparse modeling, we force this combination to have few non-zero elements.

$$error(x) = |x - y|_2^2 + |R_{ij}x - D\alpha_{ij}|_2^2 \, s.t. \, |\alpha|_0^0 < L \tag{C.2}$$

Note that all the overlapping patches are considered. Also, note that in this case the model is for the individual patches.

C.2 Gaussian Mixture Models

One very used sub-category of sparse modeling in Gaussian Mixture Models. Again, we will model each patch of the image as a linear combination of atoms in a dictionary. In the general case, supposing we have set a sparsity parameters of a maximum of L elements, and that we have K atoms in our dictionary, we have a total of $\binom{K}{L}$ subspaces that could be used to approximate a patch. So many possibilities might make it unstable - small variations in the images generate very different representations.

In GMMs, we will have a dictionary formed by a small number of Gaussians - around 10 or 20. We define our Gaussians to have the number of dimensions equal to the number of pixels in the patch - for a 8x8 patch, that will be a 64 dimensions. Then, we generate an atom of the dictionary for each of these dimensions. Now for a small number of Gaussians and a reasonable patch size, the number of combinations also grows very quickly. The main difference then, is that in this case we group the atoms in such a way to generate much less possibilities: each patch has to be represented by only one Gaussian - by a linear combination of the atoms that belong to it. That makes the number of subspaces that can be used to represent each patch decrease from $\binom{K}{L}$ to the number of Gaussians we have. Having 10-20 Gaussians already generates very good results.

APPENDIX D VÍDEOS PANORÂMICOS PARA NAVEGAÇÃO NÃO-LINEAR

D.1 Introdução

Existe um movimento por parte de grandes universidades para a divulgação de material educacional na internet, sem custos, para todos. O impacto é enorme - centros de excelência estão educando milhões de alunos ao redor do mundo.

Um estudo recente mostra que o uso de vídeos é parte fundamental da instrução dos alunos (ROSE, 2009). Vídeos tem vantagens mesmo quando comparados com aulas convencionais. Pausar, repetir, avançar - todas essas são interações que tornam a experiência mais customizável e efetiva.

Apesar dessas características, para encontrar pontos de interesse no vídeo, ainda é necessário navegá-lo usando uma barra temporal. Frequentemente, é difícil lembrar exatamente em que ponto do vídeo uma equação ou conceito foi explicado. Uma maneira natural de navegar o vídeo, sem usar a barra de tempo, é apenas indicando o conteúdo que se deseja rever. Nesse trabalhamos, propusemos uma nova interface que permite ao usuário navegar de maneira mais intuitiva. Na figura D.1, podemos ver algumas das extensões propostas por nós.

Primeiramente, criamos uma representação panorâmica dos vídeos para permitir que o usuário navegue para um ponto específico do vídeo, apenas apontando para ele. Acreditamos que esse seja o primeiro trabalho a propor esse tipo de navegação não-linear.

D.2 Sistemas de e-Learning

Para atingir o objetivo de ser aplicável em abrangência, nosso sistema precisa tratar uma variedade de vídeos. Nessa seção, discutiremos as principais classes de vídeo-aulas existentes.

D.2.1 Vídeos Baseados em Canvas

Um vídeo baseado em canvas é um vídeo cujo conteúdo é mostrado em um quadro. Os elementos do vídeo são texto e equações. Essa ideia de canvas é essencial - ela nos permite inferir a posição relativa dos frames, importante na criações do panorama. O maior representante desse tipo de vídeo é o site Khan Academy (KHAN ACADEMY, 2013).

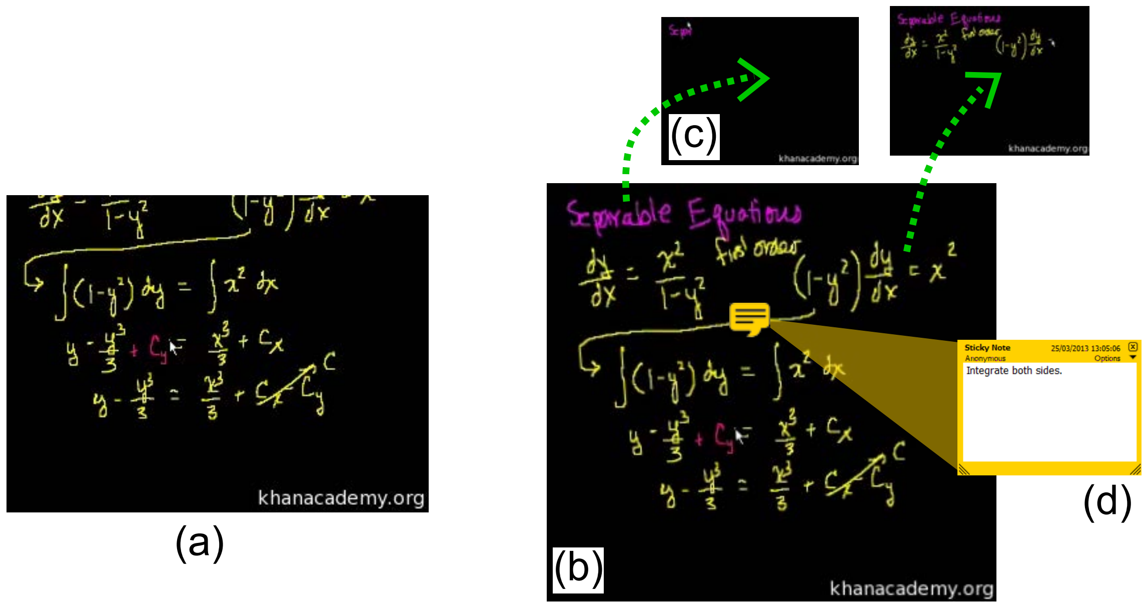


Figure D.1: Interface. (a) Frame regular no tempo 4:43 do vídeo. (b) Frame panorâmico no tempo 4:43 do vídeo. (c) Hyperlinks permitindo a navegação direta para os momentos 0:30 e 1:47 do vídeo. (d) Nota adicionada ao vídeo.

D.2.2 Vídeos Baseados em Slides

Nos vídeos baseados em slides, o fundo do vídeo é um conjunto de slides. Em cima disso, o instrutor eventualmente marca partes importantes ou faz anotações. Nesse caso, não há uma relação espacial explícita - no nosso sistema, essa informação é inserida artificialmente. Os maiores representantes desses tipods de vídeos são Coursera (COURSERA, 2013) e edX (EDX, 2013).

D.2.3 Aulas Convencionais Gravadas

Uma das maneiras mais comuns de disponibilizar vídeo aulas ainda é simplesmente gravar aulas convencionais. Esses vídeos certamente constituem o maior desafio para nossa técnica. Possuem imensa variabilidade, tornando o processamento automático uma tarefa extremamente difícil. Vídeos dessa categoria incluem o MIT OpenCourseWare e vídeos de aulas convencionais gravados por universidades que são simplesmente disponibilizados online.

D.3 Videos Panorâmicos para Navegação Não-Linear

Essa seção descreve o método proposto para criar vídeos panorâmicos que permitam a navegação não-linear. Primeiramente, discutiremos a interface proposta, com todas as extensões que queremos prover. Em seguida, discutiremos as estratégias utilizadas para tornar a implementação da interface possível.

D.3.1 Interface

A interface é a maior contribuição deste trabalho. Nessa seção vamos discutir cada extensão proposta separadamente. Nas próximas seções, discutiremos os desafios que tiveram que ser enfreantados para possibilidar a implementação dessas extensões.

Panoramas. A primeira extensão que queremos prover é a visão panorâmica da sala de aula ou do canvas onde a aula está sendo ministrada.

Geração automática de notas. O panorama mostra todo o conteúdo da vídeo-aula, sendo um bom sumário para ser exportado como um conjunto de notas. Nossa interface também permite que o usuário faça anotações e exporte para um arquivo.

Hyperlinks. Um conceito chave no nosso design é a noção de navegação não-linear através de uma interface simples. Nós queremos possibilitar ao usuário o acesso ao instante em que um tópico foi introduzido apenas clicando nele.

Extensões disponíveis em tempo real. Uma questão muito importante no nosso sistema é a performance. Dificilmente um usuário vai esperar horas para assistir um vídeo, portanto é importante que as extensões estejam disponíveis em tempo real. Isso não seria possível se nossa técnica envolvesse pré-processamento.

D.3.2 Criando Vídeos Panorâmicos

A primeira extensão essencial que provemos nos nossos vídeos são os frames panorâmicos. Para isso, é preciso registrar os frames em relação uns aos outros e encontrar uma transformação projetiva de cada frame em um sistema de coordenadas global. Em todos os casos, para a detecção de features, sua descrição e correspondência, foi usada a implementação em GPU do algoritmo SIFT (WU, 2007). Essa detecção é feita em separado para cada frame do vídeo.

D.3.3 Criando Hyperlinks

A criação de hyperlinks é a extensão mais importante da nossa interface. Como queremos que a navegação não-linear esteja disponivel em tempo real, não podemos preprocessar o video inteiro. Em vez disso, usamos os dados que já estão sendo extraídos para a criação dos panoramas. Quando um texto aparece no vídeo, ele provavelmente gera alguns pontos SIFT. Para cada local no nosso sistema de coordenada global, guardamos o momento em que uma feature apareceu ali pela primeira vez, e usamos essa informação para navegar quando o conteúdo é selecionado.

Cada vez que calculamos a transformação de um frame no panorama global, percorremos os sifts e testamos se as posicões já possuem feature. Se não, o frame atual é salvo como a primeira aparição do feature daquela região. Quando um usuário clica no mapa, a interface navega para o respectivo frame. Para evitar o aglomeramento de features, segmentamos nosso mapa clicável em regiões de 10×10 pixels, com no máximo uma feature cada. Podemos ver os SIFTs armazenados na Figura D.2.

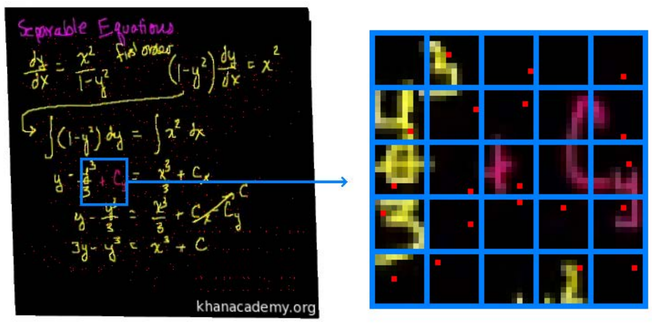


Figure D.2: Criação dos hyperlinks. Os pontos vermelhos representam SIFTs ligados a hyperlinks.

D.3.4 Geração Automática de Notas

Nosso sistema gera notas de aula automaticamente. Para isso, simplesmente exportamos o panorama juntamente com as anotações feitas pelo usuário. Para tratar arquivos PDF de maneira robusta, foi usada a biblioteca Poppler, que é facilmente integrada com o biblioteca Qt, usada para a interface.

D.4 Resultados

Essa Seção demonstra o uso da nossa técnica, aplicada aos três tipos de vídeos discutidos.

D.4.1 Panoramas

O primeiro passo no nosso pipeline é a criação de frames panorâmicos. Nós detectamos SIFTs para cada frame e achamos as correspondências, de um frame ao próximo. As homografias são estimadas de acordo com isso, e a transformação global é encontrada conectando essas projeções.

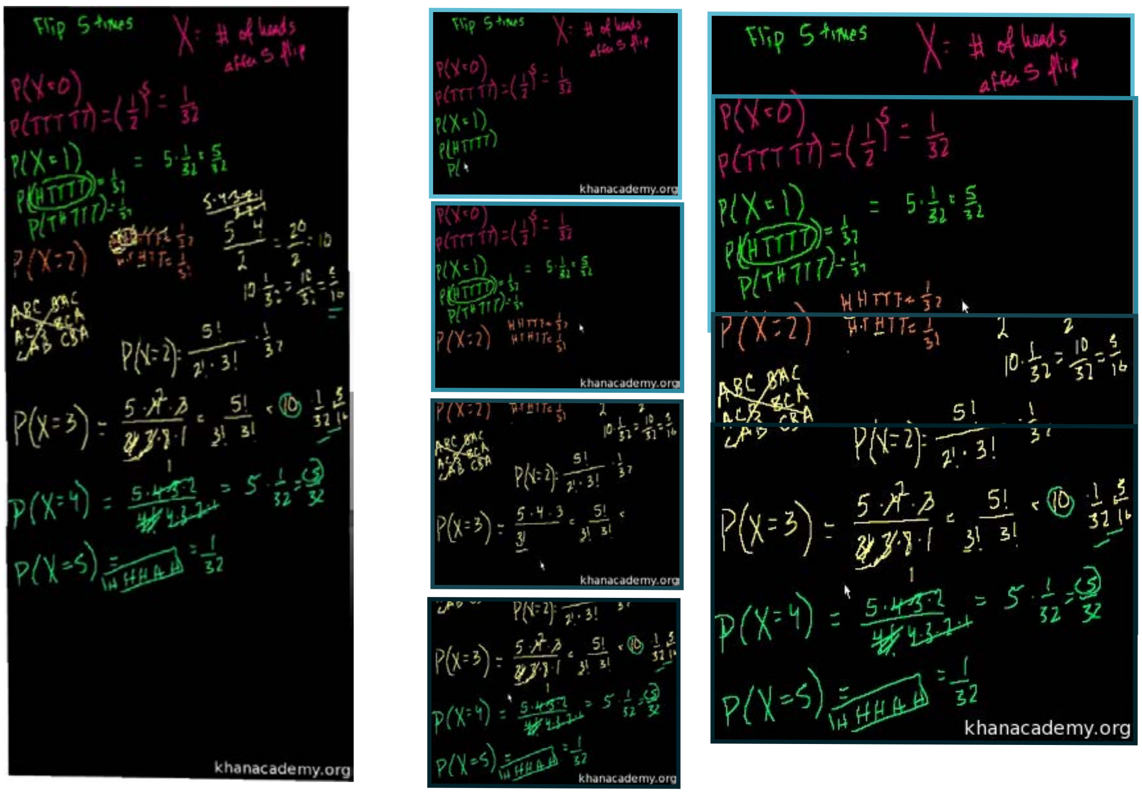


Figure D.3: Resultados do panorama - Khan Academy. (esquerda) Panorama final. (direita) Subconjunto de frames que forma o panorama.

D.5 Navegação beaseada em Conteúdo

Hyperlinks são uma propriedade dinâmica da nossa interface. Para demonstrar nossos resultadoss, capturamos alguns momentos: mostramos os panoramas e os hyperlinks, com os frames aos quais eles levam.

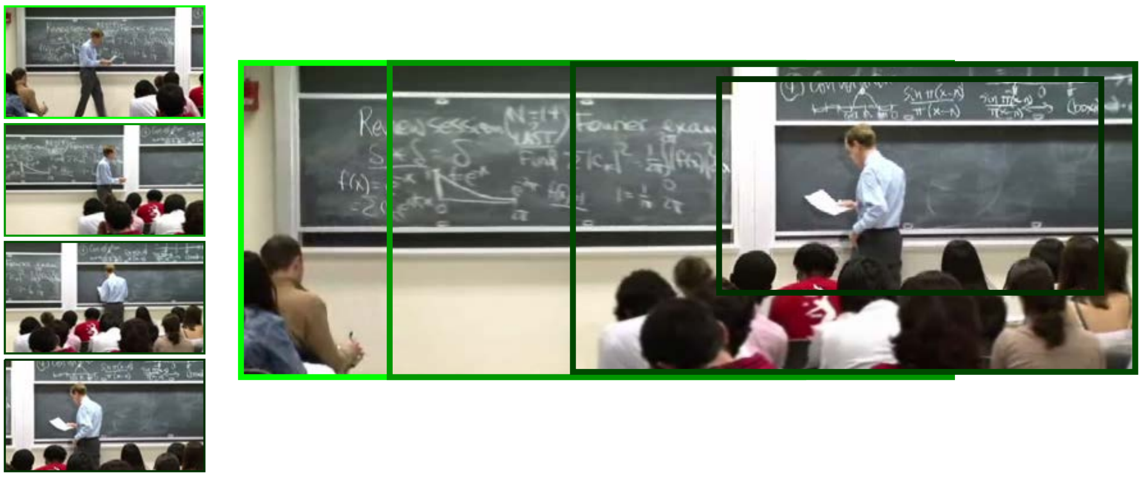


Figure D.4: Alguns frames (esquerda) usados no panorama da aula convencional (direita). Note como o sistema dá suporte ao zoom da câmera naturalmente.

Na Figura D.5 vemos um resultado de um vídeo baseado em canvas. A imagem à esquerda mostra o panorama, destacando 3 hyperlinks. As imagens àdireita mostra os frames destino correspondentes. Também destacamos a posição do cursos, para demonstrar como nossa interface navega precisamente para o momento em que o conteúdo está sendo introduzido no vídeo.

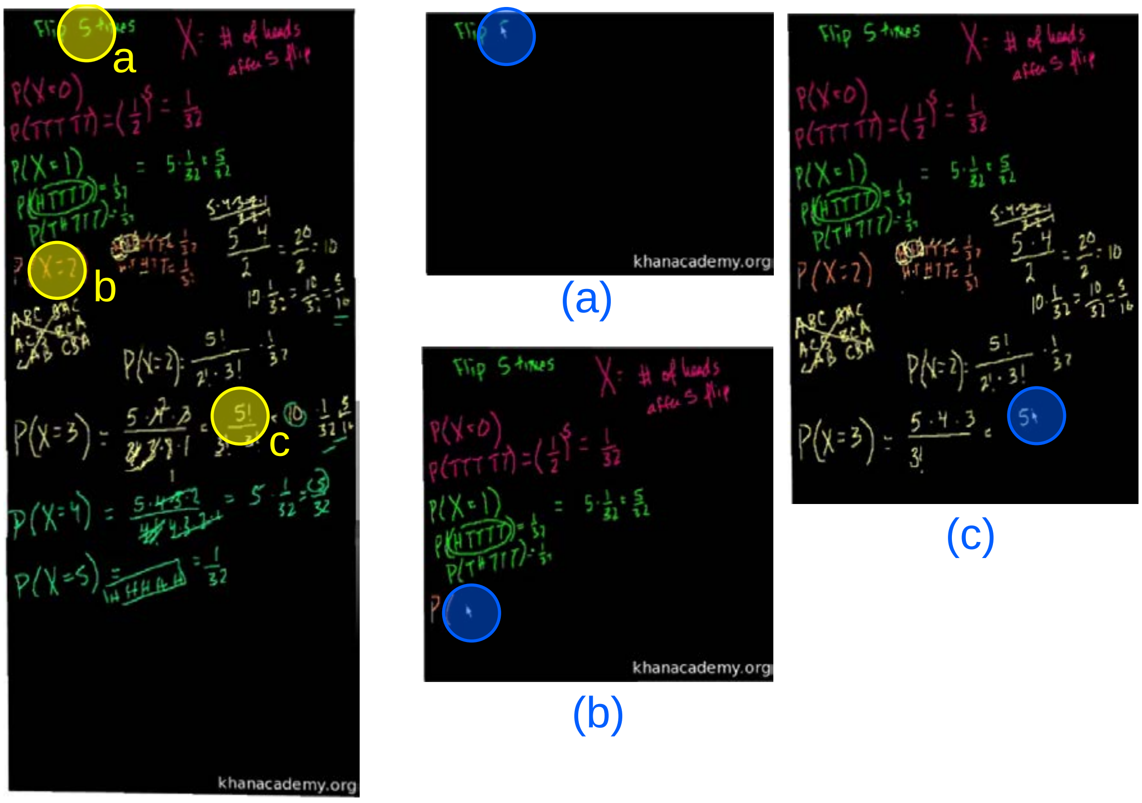
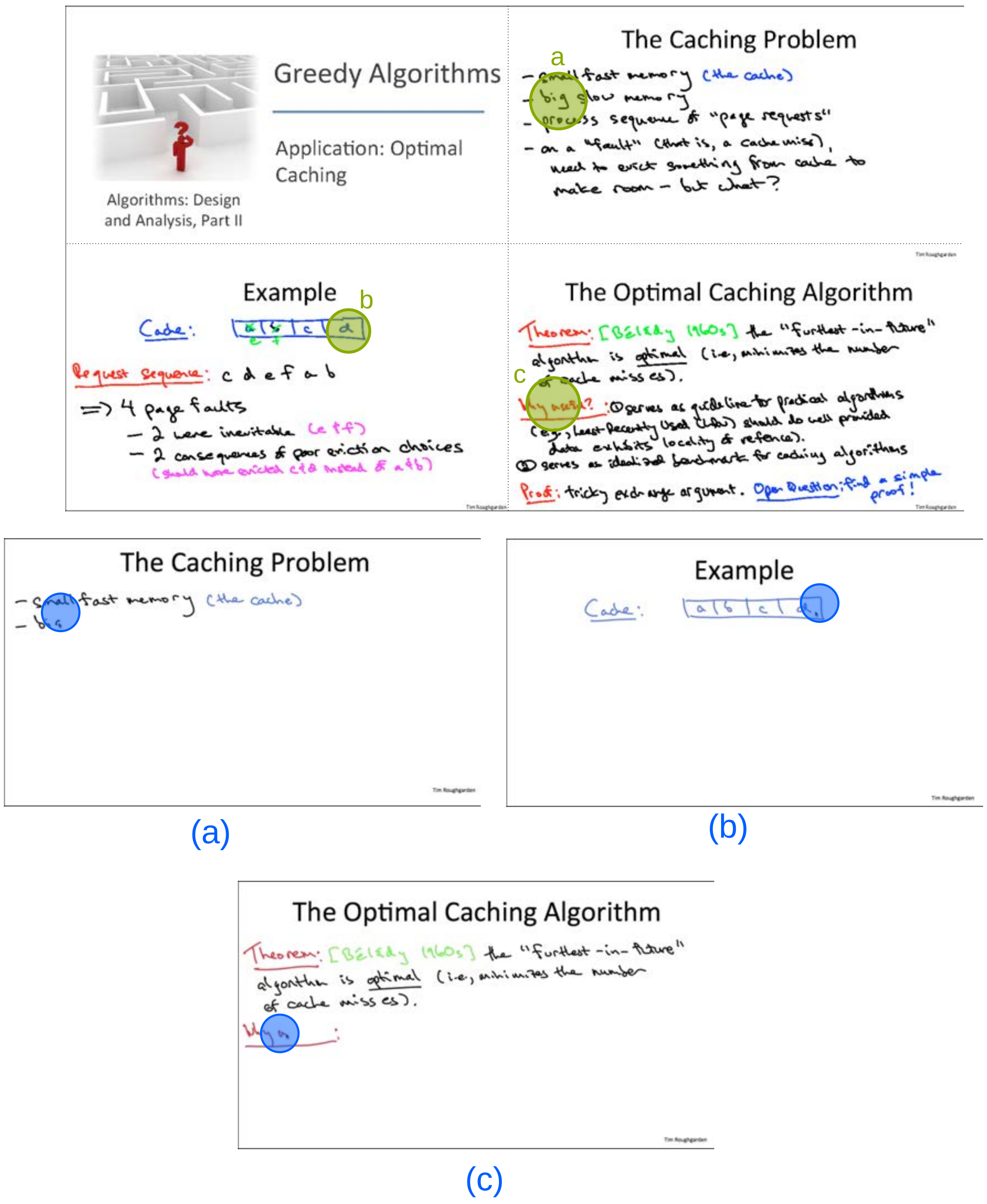


Figure D.5: Resultados de hyperlinks - Khan Academy. Panorama com alguns hyperlinks e os respectivos frames-alvo.

A Figura D.6 mostra os resultados para um vídeo baseado em slides. Como explicado, o panorama é calculado importando-se um conjunto de slides - ele é constante para todo o vídeo.



Greedy Algorithms

Application: Optimal Caching

Algorithms: Design and Analysis, Part II

The Caching Problem

- small fast memory (the cache)
- big slow memory
- process sequence of "page requests"
- on a "fault" (that is, a cache miss), need to evict something from cache to make room - but what?

Example

Cache: a b c d

Request sequence: c d e f a b

⇒ 4 page faults
- 2 were inevitable (e + f)
- 2 consequences of poor eviction choices (should have evicted c + d instead of a + b)

The Optimal Caching Algorithm

Theorem: [Bélády 1960s] the "furthest-in-future" algorithm is optimal (i.e., minimizes the number of cache misses).

Why useful?: ① serves as guideline for practical algorithms (e.g., Least Recently Used (LRU) should do well provided data exhibits locality of reference). ② serves as idealized benchmark for caching algorithms

Proof: tricky exchange argument. Open Question: find a simple proof!

Tim Roughgarden

Figure D.6: Resultados de hyperlinks - Coursera. (acima) Panorama. (abaixo) Frames destino.

O último resultado da navegação com hyperlinks é a aula convencional. Para demonstrar, utilizamos um vídeo de alta resolução. Nesse caso, não há cursor, mas podemos ver que o instrutor está prestes a escrever o conteúdo selecionado.

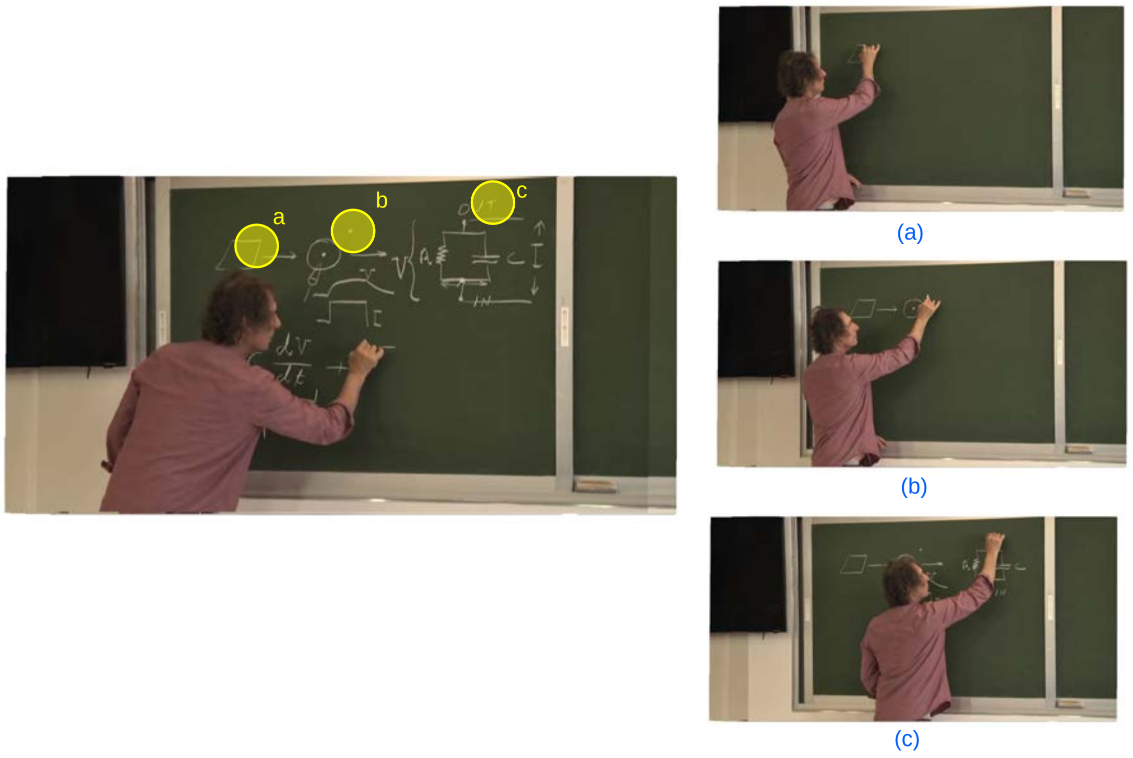


Figure D.7: Resultados para aula convencional. (esquerda) Panorama com hyperlinks destacados. (direita) Frames destino.

D.6 Notas de aula

Nossa sistema também permite a geração de notas de aula. O usuário toma as notas durante o vídeo, e elas são exportadas junto com o panorama final em um PDF.

Na Figura D.8, podemos ver dois momentos do vídeo onde o usuário está tomando notas. Esses momentos foram gravados de dentro da nossa interface.

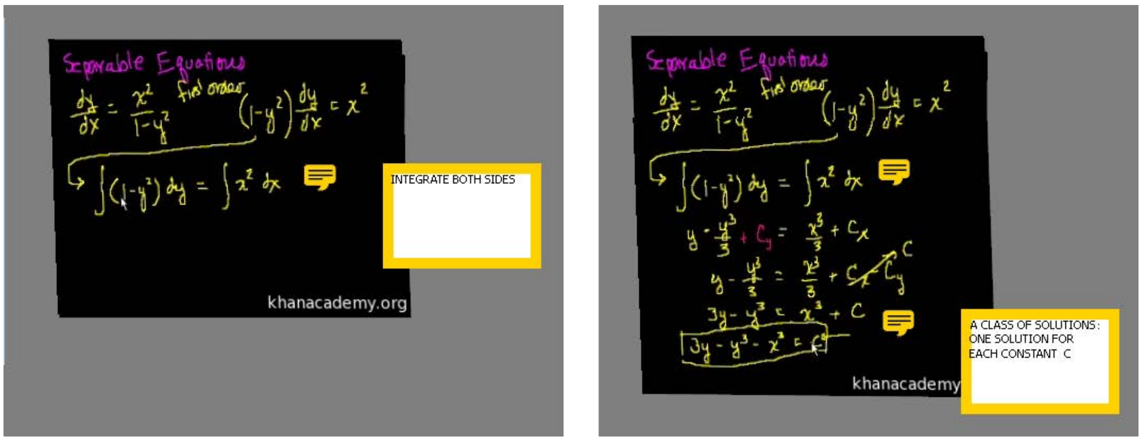


Figure D.8: Usuário pode tomar notas durante o vídeo . (a) 3min07s. (b) 5min25s.

Na Figura D.9, podes ver uma panorama com anotações dentro de um visualizador de PDFs.

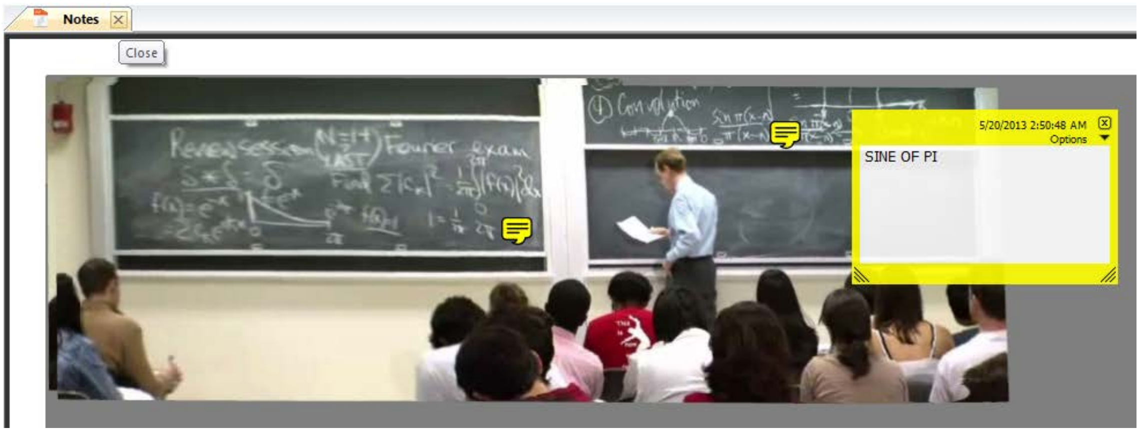


Figure D.9: Notas de aula de vídeo convencional em um visualizador de PDF.

D.7 Conclusões e Trabalhos Futuros

Essa dissertação introduziu uma nova interface para e-learning com frames panorâmicos e navegação baseada em conteúdo. Demonstramos a abrangência da nossa técnica aplicando-a a três tipos diferentes de vídeos: baseados em canvas, baseadaos em slides e aulas convencionais gravadas. Nosso sistema alcança performance de tempo real para baixas resoluções (320×240). Para resoluções mais altas, é necessário algum pré-processamento.

Nosso sistema é baseado em técnicas de Visão Computacional, e assim, herda as limitações dessas técnicas. Melhoria nos algoritmos de Visão Computacional utilizados representariam melhores resultados na nossa interface. Outro problema é que alguns SIFTs podem ser detectados como relevantes, mas serem resultado apenas de um sombreamento ou do instrutor. Nesse caso, nossa navegação linear pode se perder. Soluções para esse problemas constituem direções interessantes para exploração futura.