Rochester Institute of Technology

Department of Computer Science

MSc Project

Math Expression Retrieval Implemented Through Sketches

Presented by

Kenny Davila

Advisor:

Dr. Roger S. Gaborski

Reader:

Dr. Richard Zanibbi

Observer:

Dr. Peter Anderson

May 13, 2013

Table of Contents

1. Abstract

A collection of videos can be seen as a source of large amounts of information especially if the content of those videos are lectures. The presented project works specifically with videos from math lectures. For each lecture, two videos are provided: one captured with a camera on the classroom, and the second captured by the software of a Mimio device. Both videos record everything that the professor writes on the whiteboard. An application for extraction and retrieval of that content written on the whiteboard is presented.

However, there are different challenges involved in the process of extracting whiteboard content from the videos, and once that the content has been obtained in the form of images, it is required to compute and index different features that will describe the math that is found on those images. Queries come in the form of images of part of the content of the whiteboard as well. The proposed method, however, is recognition-free which means that no optical character recognition is performed. Then, matching of the content between a given query and the extracted images is based on visual similarity only. Compared to standard content based image retrieval, the proposed method cannot rely on features of color or texture to match visual similarity of math formulas, and for this reason sketch based image retrieval methods must be applied.

2. Introduction

A video can store an important amount of information in the form of sequences of images and audio channels. This information may be relevant to any field in general, and one example would be education if the video belongs to a math class lecture. During a math lecture, the instructor can be recorded as he or she speaks his thoughts and writes notes and formulas on the whiteboard. This record could be accessed later by students who want to watch again the explanation of a topic taught on that lecture. The problem arises when it comes to find a short explanation that may last a few seconds on a collection of videos that may last for more than one hour each. Therefore, it is important to know the content of each video and to define a way to access randomly this content as required. The main goal of the project Math Expression Retrieval Implemented Through Sketches is to develop a procedure that given a section of a frame of the video of a lecture in math, the procedure will then return a ranked set of images content related to one on the input given. Such procedure will require a way of indexing the videos in terms of key frames and features of the content, and also a search algorithm that has to be able to rank the similarity between the content of the input query and the content of any segment of video in the collection.

A procedure for indexation and retrieval of videos of math lectures could easily become part of a larger project designed to help students in general. Moreover, students with special disabilities could become the more benefited from the application of an algorithm of this kind. For example, imagine the case of a student that is visually impaired, but is still able to use an iPad during a math lecture. He or she might not be able to see directly what is written on the board, but could be able to see it indirectly using an iPad application that has access to the video of a camera recording the board. Searching information taught on previous lectures is a difficult task for students with this kind of disabilities, but the difficulty could be greatly reduced using a system that applies video search. The student can take a snapshot of the video of the board and use it to query information from past lectures in a very fast and convenient way. With all this information available in real-time, the student could be able to keep up with the rest of the class. Of course, this kind of search would be helpful not only in the classroom but out of the classroom while studying and looking for specific content.

However, the problem of searching for related videos based on the content can be classified into many categories except in the category of trivial problems. Basically, many video search algorithms are based on labels of the videos given by the users of the system, but this is not the current case. In the current problem, the user must be able to search for anything written in the content of the board, and it would be a long task for a human to label a segment of video with all the possible tags for the current content. What is more, standard systems based on labels do not consider any kind of spatial relationship between each individual label. For example, consider the case of figure 1, where two given numbers are given, and while tags can find these numbers, they might not be related in the expected way. It could be argued that a more complete labeling system would include these relationships as part of the tags, or even more, than LaTex or any other string representation of these formulas can be used as a tag. But the main problem persists where a human would be required to manually tag those formulas or a system

with a high level of accuracy for formula extraction would have to be applied. The proposed approach tries to be as automatic as possible by eliminating the need for human labeled data humans on its different subcomponents and algorithms.

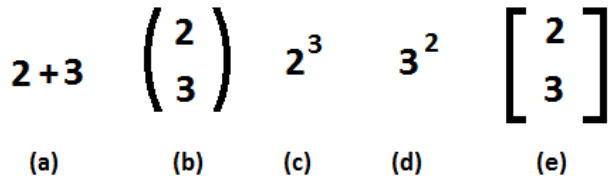


Figure 1. A few images containing 2 and 3.
(a) On a sum
(b) Combination
(c) Power
(d) Power
(e) Column vector

The largest category where the proposed search procedure fits is into the category of information retrieval (IR) which deals with search of information inside of large collections of data. More specifically, the problem belongs to the subcategory of content-based video retrieval (CBVR) because the information to search for is stored in the form of video files. Since video is composed of both audio and images, usually a multi-modal search procedure could be applied, but in this case, the proposed search procedure will use only image information to describe the content of the video, and therefore it could be counted as a content-based image retrieval (CBIR) procedure. However, standard CBIR procedures use color and texture to describe the content of an image, and such information is not relevant for this application since the same content can be written using different colors, and it should be matched regardless the color used in the query and the color found on the videos. In addition, the important content of the images is mainly text, math formulas, and diagrams which makes the problem related to those found in many document image retrieval (DIR) applications, but given that most of the content belongs to the field of math, and it is composed of a high number of explicit math formulas, it would fit better in the subcategory of math recognition (MR). However, the proposed approach is recognition-free because it does not apply explicit optical character recognition (OCR) in the way that a standard MR application would do.

One of the most important differences between the proposed application and many of the DIR applications is the fact that input images belong to handwritten text and handwritten math expressions instead of scanned images of printed text. What is more, as it has been described in the work by Liwicki and Bunke [1], handwriting on a whiteboard tends to be different compared to handwriting on paper or pen-based computer because the writer usually needs to stand in front of the board, and the patterns of the writing of a person in such condition are quite different to those of a person writing while sitting on a chair. Also, another important difference is that content can be written in many ways as there are no predefined layouts. With so much freedom for drawing on the whiteboard, and content that can be really varied, the proposed approach belongs to the category of sketch-based image retrieval (SBIR), where the content on a video of the whiteboard can be described as a sequence of sketches, and the input query is a sketch for which the most similar sketches need to be found.

To start with a video file and to end with a database containing an index of sketches suitable for fast retrieval requires a processes pipeline composed of a few complex sub processes. Figure 2 shows a summary of the indexation pipeline of the proposed method. The first the sub processes in the

indexation pipeline is the one in charge of taking the input videos and detecting specific frames and regions of interest where the sketches will be extracted from. The criterion used to select each candidate region of interest is based on the changes made to the content of the whiteboard in the videos. This partitioning of the content of the video is not a trivial task, and it should be mentioned that the current input for the system will consist of two separated videos, the first of them belongs to a still camera located on a classroom recording the content of an entire whiteboard, and the second video belongs to the sequence of strokes captured by a MIMIO capture device. This second video has the advantage that only the strokes are recorded, but currently has a great disadvantage in terms of a reduced video resolution and noise. More details about these input videos will be given in section 3. Since two videos of the same content are given as inputs, a matching procedure is required that will identify the visual correspondence between the videos, and also will determine an offset of time to synchronize the videos because these are not guaranteed to start exactly on the same millisecond.

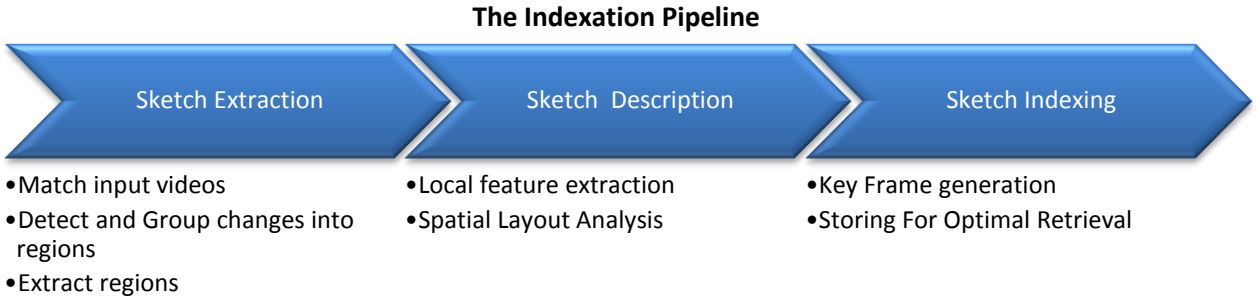


Figure 2. Outline for the Indexation pipeline

The second sub process in the indexing pipeline of the system is the content description generator that works for a given region of a frame. This description is important because the accuracy for the matching procedures is directly linked with the selection of features used to describe the regions. As mentioned before, the system is based on SBIR which means that the system treats the individual regions as hand drawn sketches, and consequently the descriptions used are based on the ones typically used for SBIR. A sketch is usually described at two levels: global level and local level. The first one, global level, provides general features of the sketch mainly related to the spatial layout. The second one, local level, provides specific features for each individual primitive of the sketch. In this case, these individual primitives are the connected components of the image. Local descriptions have to be generated for each of these connected components, and the result of this description will be in the form of a feature vector. While the system intends to be recognition free, features that are typically used for offline OCR can be used to describe the individual connected components in a way that two components should be considered similar if they also represent similar characters. Nevertheless, not only characters can be found as connected components, as there can be anything drawn on the board, and for this reason the local description also needs to be as generic as possible. Finally, the spatial layout between components is generally represent as a topology graph where each connected component is represented by node and each edge between two nodes is equivalent to some kind of spatial relation existing between these components. Because many types of spatial relations can be considered in the

model, more than one topology graph can be built to describe the sketch, one per each kind of spatial relation considered. In the actual implementation only the neighbor relationship is considered.

The third and last sub process of the indexing pipeline is the indexing component itself. This component will take the description of each sketch and will store this information in a way that it can be easily retrieved in the future. This part is also in charge of generating groups of sketches based on their time stamps to form key frames that can be used to describe the content of the entire video in just a few images. One of the main challenges for the indexing component is that given the description of a sketch query, the system has to be able to retrieve the most similar sketches in a fast and reliable way. The complicated part of this fast retrieval process is the fact that descriptions of sketches are complex and it usually requires some heavy processing to determine the exact similarity between two given sketches. If there are thousands of sketches on the database, and the time required to compare the similarity between two sketches is exactly one second, then comparing the input query with all the sketches stored in the database will take several minutes or even hours. For this reason, an ideal indexing procedure must be able to make a fast estimation of the similarity between a given sketch and several sketches in the database, and then retrieve only those ones that are worth the time required for a more detailed comparison. In section 2, some methods used for indexing will be described, and it will be shown how a common factor between these methods is that they are directly related to the selected measurement of similarity between sketches. This makes sense because different similarity measurement approaches will depend on different features to define when two sketches are very similar, and consequently the indexing procedure must capture the most heavily weighted features and use them to make this fast similarity estimation. Because of time restrictions, the final index used in the current implementation does not apply any kind of speed up based on fast similarity testing, and while optimizations of this kind are left for future work, the current system does create an index file with pre computed elements that if it was not there, the whole retrieval process would take several minutes for every input query.

The indexation pipeline represents all the preprocessing required to convert input videos into something that can be search and retrieved. The second main component of the project is the search and retrieval procedure. To start with a region of an image and to end with a ranked list of similar segments of video also requires a small pipeline of sub processes which is outlined in Figure 3. The first process required is the content description generator, the same one used to generate the descriptions of the sketches during the preprocessing phase. After the description of the input query has been generated, the next step is to find similar sketches on the index created during the preprocessing phase. Since no special method for fast testing of similarity was implemented on the current scope, the system will apply a similarity measurement procedure between the input query and all the sketches stored on the index, and will use the resulting score to rank them by similarity and finally retrieve only the top N matches. The similarity measurement procedure is probably the most important element of the current project. Section 2 will describe different approaches that have been used for measurement of similarity between sketches. Some of these methods are easier to implement and run in faster time, but might produce more false positives, while other methods have better support for partial matching and will produce a more reliable measurement of similarity, but their time complexity is too high to be applied

over sketches with more than a hundred connected components. Partial matching refers to the case when a formula might have a very similar structure to another but a few components might be missing or be different between the query and the best candidate of the stored sketches, and if this is the case, the system should still consider this candidate as a strong one with just a small penalty for the missing parts. In practice, most of the related sketches are partial matches of content instead of complete perfect matches.

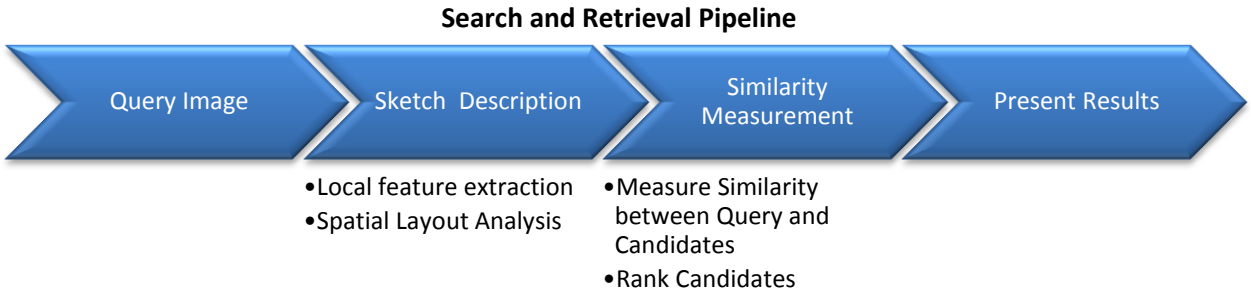


Figure 3. Outline for the Search and Retrieval pipeline.

One of the main goals of the current project was to implement at least two different procedures of similarity measurement, and then select the procedure that offers the best trade-off between quality of retrieval results and running time, and finally to suggest and apply possible improvements to speed-up the retrieval processed and/or increase the quality of the search results. In those terms, five different similarity measurements were tested and one of them, recall of matched pairs of the neighbor graph, was the one that produced the most satisfactory results. It is worth to mention that improvement of quality of the results can be obtained through relevance feedback. However, relevance feedback requires a high level of interaction with the user, and depending on the measure of similarity, it requires applying certain changes on the measurement of similarity or even in the query in order to make the feedback effective. However, due to time restrictions relevance feedback has not been implemented on the current project.

3. Background

The problem of finding specific information within videos of presentations accompanied with whiteboard handwritten notes has been studied before for different applications. One example are the four systems developed in the work by Marcus Liwicki and Horse Bunke [1], which also explains that these problems are multi-modal and need different algorithms for recognition at the level of speech, handwriting strokes, and images from the video in order to get the maximum possible accuracy on a search procedure. The proposed approach works only with the images extracted from the videos, but it still requires of different algorithms in order to implement its two main pipelines. None of the required algorithms represents a new problem, and some research has been done for each of them. This section contains a brief summary of some approaches used for each problem, and pays special attention to those that will be used as a base for the implementation of the current project.

3.1 Change Detection and Automatic Key Frame Extraction on videos

Probably the first problem that needs a solution is the automated detection and extraction of key frames from the videos of math lectures. The general problem of automatic key frame extraction has been heavily studied for different applications and different solutions have been proposed, but only a few of them are analyzed here. A key frame is a frame that can be used to describe an entire segment of a video, and therefore a subdivision of the original video file into segments has to be done in order to select these key frames. These subdivisions are usually created based on automated shot detection which can be done by calculating differences between continuous frames and applying thresholds to these differences. The approach by Calic and Izquierdo [2] uses statistics of the macro-blocks descriptors in the MPEG standard to find patterns of change over the data of a compressed video for real time key frame extraction. The approach by Mohanta et. al [3] uses various features and when the majority of them reflect an important change in the content of a video, then a change of shot is automatically assumed. Guang-sheng proposes a method [4] where the images of the video are divided into a grid, and then the amount of change is calculated for each cell of the grid. A model of visual attention is applied to estimate the amount of attention that each cell could get from a viewer, and this level of attention gives each individual cell a weight. The weighted change is then calculated for the entire frame, and when it surpasses a certain threshold then a new shot is assumed. Min et. al [5] use histograms of edges and colors as features, and find an adaptive threshold for theses feature to partition the video into segments.

From the analysis of the works like [2], [3], [4], and [5], it is easy to conclude that methods based on shoots and major changes in the video will not work for videos of math lectures which basically consist of a single shot of a whiteboard with a speaker writing on it for several minutes. Therefore, the selected approach has to be designed for this specific kind of videos in order to be effective. What is more, not only key frames of the video have to be extracted, but the specific region changed between

each pair of key frames needs to be identified. In this sense, only applications that work with similar data to the one used on this project will provide helpful information. For example, Talkminer [6] is a system that works with webcasts of lectures. They assume that these videos contain slides, and for that reason they designed a key frame detection system that finds candidate frames where a new slide is shown on the video. This approach is close to what is needed for the current project, but it was designed to work with data with higher levels of variation, and therefore a simpler approach focused on whiteboard videos might be a better fit.

The same company that created Talkminer in 2010 also designed in 2009 a system called ReBoard [7], which is collaborative software to automatically index changes on the content of whiteboards in an office environment. The application is constantly capturing images of the whiteboard, detecting objects in motion as a way to estimate when the content is about to change. After the content is assumed to be changed, the system captures several images until it finally obtains one where the whiteboard is assumed to be free of obstructions. Given this obstruction-free image, and the previous key frame, the system can calculate the areas of the board that were changed. The image of the board is smoothed to avoid false positives on change detection caused by noisy pixels, and the detection of changes is made using two different resolutions of the whiteboard. In fact, this application does apply an algorithm with an output just as required in the current project, but the specific details required for its implementation are missing on their paper.

Another system that works with whiteboard images is Thor [8] developed at Princeton University. In the Thor system, changes are detected at the pixel level, and pixels are classified between stroke and whiteboard. The neighborhood of a stroke pixel is used to confirm that it is indeed a stroke and not an isolated noisy pixel. A changed pixel can mean one of two things: a stroke was added or a stroke was erased. The reason of why the Thor system considers the changes made at the level of individual pixels is because it uses each pixel to make an estimation of the drawing strokes on its indexing procedure of the content of the whiteboard. One drawback of this approach is that it requires a process of 24 hours of calibration to train the system with the expected ranges of color for the whiteboard pixels under different lighting conditions.

Finally, the whiteboard capture system [9] developed by Microsoft uses an approach that can be easily implemented for the current input data. Similarly to ReBoard [7], the system by Microsoft divides the whiteboard image using a grid, and changes are detected at the level of grid cells. The system uses an estimation of the whiteboard color based on the predominant color on each cell. Also, similar to Thor [8], each cell is classified between stroke and whiteboard, but an additional class is added to account for obstructions on the image. Spatial and temporal information is used to refine the classification results and separate stroke cells from obstructed cells, especially based on the fact that it is not possible for an important stroke to be on the whiteboard on isolation or for less than a second, and given this assumption the system will assume cells that became stroke cells for a short period of time to belong to the obstructed cells class. The final algorithm implemented on the project does change detection using a method with a grid and cells very similar to the method detailed in [9], but the major differences come from the fact that change detection in this project is done over a video that does not contain obstructions of the content on the whiteboard.

3.2 Content-Based Image Retrieval

The problem of retrieving images from large datasets based on their content has been studied for a while now. Many approaches have been tested and results are varied depending on the restrictions of the data being retrieved. Some approaches are based on descriptions of the dominant colors on the images as for example the work by Arjunan et. al [10] which is based on color histograms on HSV space only. Other approaches consider the fact that even the same color might represent two or more different objects and therefore texture must be taken into account. For example, the work by Chiu et. Al [11] combines fuzzy logic to assign linguistic terms like low, high, very high to the Tamura features of texture: coarseness, contrast, directionality, line-likeness, regularity, and roughness. For some applications, there might be many images representing exactly the same object but rotated, translated or scaled. To achieve a certain level of matching for that kind of applications, the scale invariant feature transform (SIFT) [12] have been proposed and used by other authors as for example in [13] where several pictures of a specific building are found using a picture of the building as the input query.

More sophisticated approaches for CBIR consider that it is important to match not only general features of the image but also specific information about the objects found on it. However, identifying all the concrete objects inside of an image is a challenging task, and for that reason some approaches based on segmentation consider the regions obtained as blobs and not as concrete objects. The work by Carson et. al [14] applies a segmentation algorithm first to obtain blobs, and then for each blob the color and texture features are calculated. Matching is done by finding images that contain blobs with similar characteristics to those found in the query image. Other approaches try to go a step further by automated labeling of images based on features of its sub-regions, like for example the work by Jin et. al [15] which applies segmentation, and with a few labeled regions on some images and using clustering, it automatically assigns labels to each region of the image based on similarity in color and texture to the examples given.

Just a few approaches have been mentioned here, but a more exhaustive list of approaches can be found in the survey by Datta et. al [16]. In fact, according to the definitions given on the survey, from the point of view of the user, the current application has a query modality based on images, the data scope is domain-specific and the user intent is that of a searcher. From the system perspective, the current application has a content-based query processing, with relevance ordered visualization and domain-specific data scope. In terms of image signature, it is a region-based signature composed by a set of vectors that describe each individual primitive and some kind of optimal matching procedure has to be applied for measurement of the similarity between images. A huge difference between the current application and many general CBIR applications is that for the current system texture and color are almost useless for the generation of image signatures, and that is the reason of why the proposed approach is mainly based on SBIR and DIR instead of general CBIR.

3.3 Document Image Retrieval and Classification

Retrieving specific images of documents from large collections is a problem that has been studied by many researchers. While the most simplistic approaches would rely on human-labeled images, many approaches have been proposed requiring a minimal human interaction. Some of these approaches would require an OCR algorithm to recognize the text before doing any kind of retrieval, but other are considered recognition-free because they are based only on features of the content without doing explicit OCR. Since the current application intends to be recognition-free, most of the approaches analyzed on this section are recognition-free as well. Two general problems exist on this area: document retrieval and document classification. Document classification deals with assigning a class or type of document to a given document image, while document retrieval consists on finding specific documents based on their layout and/or content.

In terms of document image classification, the usual goal is to characterize a document and use this description to classify between a predefined set of types of documents. A few of these approaches for document image classification are purely based on layout descriptions. This kind of approach usually provides interesting algorithms to analyze and describe the layout of a free-form document image. Modified X-Y trees are used in the work by Cesarini et. al [17] to describe the layout of text-blocks found by an external OCR in a document image, and then these trees are encoded into fixed-length vectors by summarizing patterns found on them. These encoded vectors are used as inputs of a multi-layer perceptron (MLP) that classifies the page as one of 5 possible classes. In the work by Gordo et. Al [18], an interesting cyclic polar representation of page layout is used. This representation is translation invariant and can be scale invariant if the distances are normalized using the size of the page. It is important to consider different layout representations because these provide alternatives that could be used to describe the layout of the math formulas during the indexation and retrieval processes of the current project.

Other approaches for document image classification are based on semantics of the content in the documents. While this task seems to require OCR to find common frequent words between documents, recognition-free approaches have been proposed to solve this task. In the work by Barbu et. al [19], a system of “bags of symbols” is used where each connected component is described by a set of features, and then clustering is applied to group similar connected components assigning nominal labels to each cluster. Then, each page is represented by a graph where nodes correspond to labeled connected component and edges are added based on proximity between connected components. Graph mining techniques are applied to find common sub-graphs on graphs from the collection of documents. The common sub-graphs founds are considered symbols and the measurement of distance between two documents is given in terms of common symbols. A support vector classifier is then used for the final classification of documents in the collection.

In terms of DIR, the usual goal is to retrieve pages of documents based on its content, but it can also be based on the layout of the document. The work by Hu et. al [20] proposes a method for DIR where each page in the document is divided into a grid, and then each cell is catalogued as text cell or

blank space based on its black pixel density. A distance metric is provided to compare grids of different pages, and this metric is used to rank similarity between an input page and other pages stored in the database. The system will then return the pages with the most similar layout. Similarly, the work by van Beusekom et. al [21] uses abstractions of the text paragraphs by representing them as blocks. Different measurements of distance based on position, area and dimensions are provided to compare the similarity between two blocks of two different pages. Given the distance between all pair of blocks, a final assignment between pairs of blocks is made trying to minimize the sum of all distances. The final distance can be used to rank similarity between one page and several pages in the database, and finally the most similar are retrieved. However, while two pages can have exactly the same layout, their content might be completely different, and a document with exactly the same content might be represented using a completely different layout. For the context of math formulas, spatial layout between elements is important, but the specific elements present on the formula are important as well, and for this reason an approach purely based on layout cannot be applied.

Different approaches have been proposed for recognition-free DIR based on the content of documents. In the work by Rath et. al [22], a word-spotting approach is proposed which uses ink projection profiles to describe individual words, and then applying dynamic time warping (DTW) the system estimates the minimum distance between the profiles of two words. Using this measurement of distance between pairs of words, the system can cluster words found in historical handwritten documents for automatic indexation. Labels can be assigned to each cluster for later retrieval of these documents using keywords. Li et. al [23] proposes a method for word encoding for printed text which allows to represent individual words in terms of coded strokes. With this coding system, is possible to find keywords on documents and do retrieval based on keywords. A major drawback of the system of encoding used is that it works with a limited set of printed fonts only, and it will not work with italics. An improved version of this method is presented in [24], where the authors modified the encoding system to make it more robust than their previous work, but it is still designed to work with images of printed documents.

The current discussion of works is by no means exhaustive and more detailed information containing a larger list of approaches used for document image retrieval and classification can be found in the survey by Marinai et. al [25]. What can be concluded from the analysis of these works is that DIR is closer to provide useful solutions for the individual problems found in the current application than general CBIR. However, many of the general DIR approaches are designed to work with printed documents images, and many of them are designed for text-only documents. Given that an image of the content of the whiteboard contains text paragraphs only, it is possible to apply a word-spotting approach [22] to retrieve other images that contain many common words. However, it is not easy for the application to decide if a given image contains text only or math formulas only, and for to accomplish this task an approach for region classification between text and math formulas would be required.

3.4 Identification of math regions in text documents

As mentioned before, an algorithm that helps to distinguish between text regions and math regions would be really useful in the current application. This problem has been studied by some researchers before under the contexts of document image retrieval and math recognition. Standard OCR systems can achieve a great accuracy over text regions, but when the text contains math formulas their performance usually decays considerably [26]. This effect might be due to typical segmentation algorithms that assume that the text to recognize is organized in text columns, paragraphs, and lines which is not the case for math formulas. A common motivation to classify regions between text and formulas is to use an OCR system on text regions and a different approach for math recognition on math regions [26].

Kacem et. al [26] identify two different kind of math formulas that can be found in documents: Isolated formulas and embedded formulas. An isolated formula is a math formula that usually appears centered, on its own line and surrounded by considerable blank space. Embedded formulas are the math formulas that appear inside of text lines. The separation of math formulas from text is done in two steps: a global step that separates text paragraphs from isolated formulas, and a local step that separates words from embedded formulas in text lines. A set of symbols frequently found in math formulas is defined as possible labels, and each connected component is labeled using fuzzy memberships for each label. Text lines are segmented and then the isolated formulas are identified using certain features of each text line. For embedded formulas, a second labeling process is required that will identify the spatial relationship between connected components of the same text-line, and based on these then the system can separate possible embedded formulas from words.

In the work by Drake et. all [27], a neighbor graph is used to describe all text-lines found in a document image. On this graph, nodes correspond to connected components and edges between nodes reflect that these components are neighbors on the same text-line. Then, a set of features is extracted for each node and for each edge. With these features, two separate classifiers are used, one to classify each vertex as math vertex or text vertex using a total of 77 input features, and the second one to classify each edge as math edge or text edge using a total of 29 input features. A third classifier is then applied to combine these results and yield a final classification for the entire text-line as either math formula or text line. Note that the vertex classifier is not a symbol classifier, it just outputs whether a given connected component might be part of a math symbol or not based on its features.

Garain [29] also makes the distinction between isolated formulas and embedded formulas, and proposes a method for each. In the case of isolated formulas, these are identified using some basic features: white space surrounding the line, scattered-ness of the symbols in the line, relative height and occurrences of selected math operators in the text-lines. For embedded formulas, the system also uses some features for classification: confidence of OCR for recognized words, inclination of a sentence to contain words likely to be math formulas, type style of words, scattered-ness of symbols in a word, and

inter-character distance within the word. Note that a major requirement of this method is the assistance of an external OCR system.

Methods like the one by Kacem et. al [26] and Drake et. al [27] could work for the current application. In general, it is desirable for the final approach to be recognition-free in a way it does not need to use an external OCR like the method by Garain [29]. Also, it is important to note that these method as well a few others that might have been proposed are mainly designed for images of printed documents and not for handwritten symbols which means that some adaptions would be required to make them work with images of handwriting on a whiteboard. Because of restrictions of time, no classification between math regions and text regions was applied on the current system.

3.5 Off-line Math Recognition and Retrieval

The field of off-line math recognition corresponds to algorithms that can extract math structure from 2D images. There are several challenges associated with this problem, especially when automated evaluation of expressions is required. Probably, one of the most difficult challenges is being able to capture the entire hierarchy implied by the relative location of operators and operands in a math formula. Spatial hierarchy of elements in math is really important, and the current system could try to capture such hierarchy for later similarity matching, and to make this possible, similar methods to the ones used to capture the hierarchy of operations in off-line math recognition should be used. Moreover, the current application could be extended to apply math recognition, and do retrieval based on estimation of similarity between two math expressions, but for the current scope the application will try to find similar expressions without explicitly recognizing them specially because text and math are mixed and to apply math recognition, the text would need to be detected and filtered first.

In the work by Ha et. al [29], a method for math expression understanding is presented. This system starts with the connected component of the image, and then with these elements a process with two phases is executed to create the expression tree that would represent all elements with their corresponding operator hierarchy. The first phase is a top-down sub-process that builds an initial tree using a recursive X-Y cutting procedure based on the bounding boxes of each element. The resulting tree captures the more general relationships between elements, but a second phase bottom up is required to correctly represent the hierarchy between immediate neighbors and to respect the order of evaluation of operators.

A method for formula recognition for a large collection of mathematical literatures is presented in the work by Ashida et. al [30]. Given the bounding box that contains a math formula, the connected components are labeled, and the symbols are normalized to a predefined resolution. Four different features are used for classification of each symbol. Because their method is designed to work with off-line data, it might be the case that two or more symbols are touching each other, and one single connected component might belong to more than one symbol. A procedure to detect and correct these touching symbols through morphological operations is applied. The hierarchy of the math formula is

then extracted based on the recursive application of a set of heuristics rules based on bounding box and relative positioning between elements. What is interesting about this method is the fact that it has been tested with large collections of papers and has a really high accuracy for recognition of the spatial relationships.

A recognition-free method for math retrieval from handwritten queries is presented in the work by Zanibbi and Yu [31]. X-Y trees are used to represent the hierarchy between elements in whole regions of pages, but this method measure similarity between regions using visual similarity instead of hierarchy based. The visual similarity is measured using a method similar to word-spotting [22] which is based on Dynamic time warping of vertical projections of pixels, but in this case two vertical projections are used: one for the top half and one for the bottom half. However, if the results of this method are similar to the ones in word-spotting [22] and structural similarity is not very important, then two large expressions with the same structure but using different operands (numbers and/or constants) would not be considered similar as they should for current application.

Only a few approaches were mentioned on this section but an exhaustive analysis of works published on this area of math recognition and retrieval can be found in the survey by Zanibbi and Blostein [32]. As it was mentioned before, the current application is intended to be recognition-free, and only recognition-free methods of math retrieval can be considered for implementation. Also, the system needs to be able to work with handwritten formulas, and also it has to work with many kinds of mathematical expressions, especially with vectors and matrices for linear algebra courses. The final approach could be a hybrid taking ideas from the fields of math recognition for formulas, DIR for text, and sketch retrieval for other kind of elements that could be drawn on the whiteboard. However, as it was stated before, because of time restrictions no math recognition was performed on the current project.

3.6 Sketch-Based Image Retrieval

In the field of SBIR the usual goal is to retrieve images using hand-drawn sketches as input queries, and the images to retrieve in many cases might be hand-drawn sketches as well. Very different kinds of approaches have been tested on this field, but a common idea across many of them is the fact that sketches are build using a set of primitive elements with a certain hierarchical or spatial relationship that gives them a specific meaning. Special attention is given to this field since the developed system is based on SBIR. This subsection presents some approaches in the field of SBIR and similar applications.

There are a few specific domains where methods similar to SBIR have been applied. Usually these domains include drawings with certain structural restrictions. One example is the case of recognition of handwritten chemical diagrams [33]. A chemical diagram is intended to represent a specific chemical compound, and therefore the spatial relationship between elements used and even the types of lines have specific meanings in that field. Also, many domain restrictions will apply since not all possible combinations of symbols are legal. The approach presented in [33] for recognition of handwritten chemical diagrams relies on certain processes made on-line as the writer is creating the

diagram which is something that cannot be done on the current approach without possible loss of accuracy on trying to approximate on-line strokes from the videos. Another example of applications of SBIR with predefined domain is the recognition of handwritten concept maps [34]. On this approach the strokes are first classified between labels and content vertices. The structure of concept map is already similar to that of a graph where nodes represent concepts and edges represent the connections between concepts. If the strokes representing a concept node are correctly identified and the text is recognized with a high accuracy, it is possible to make an index of concept maps for further retrieval.

Ideally, using an approach with restrictions of structures and symbols would be able to reach higher accuracy in the domain of math expression retrieval. However, in the current application, the content written on the board can be of many kinds, and it is never restricted to specific formulas as there can be literally anything drawn on the board. Therefore, more general approaches for SBIR are needed which are able to adapt to any unrestraint structure of the input drawing and can provide ways to consider two drawings similar if and only if they have a similar structure. One example of a method for SBIR without restrictions of layout is the work by Sciascio et. al [35] which uses what they called modified Θ R-Strings to represent a given drawing by listing the description of its elements in counter clock-wise order as viewed from the center of mass of the sketch. However, this representation of structure does not seem to be the most adequate when describing math formulas with very complex structure since it is based on angles only and it losses the relationships between pairs of primitives, and therefore other possible representations based on graphs might be more adequate. Other systems for more general SBIR have been designed without specific restrictions of domain, and some of them can be easily adapted to work in specific domains like in this case math expression retrieval.

In the Thor system [8], a method for indexation and later retrieval of whiteboard content is presented. A very complete preprocessing is applied to make an estimation of the strokes used to draw each connected component in the whiteboard image. Once that the strokes have been obtained, the system generates a description for each stroke based on histograms of distances between random points, square root of the area of the triangle formed by three random points, and angle between three random points. Random combinations of three points are generated and then these features are calculated for each combination to build a distribution that, according to the authors, is stable if thousands of random combinations are used and it also will be similar between strokes with a similar shape. Additional features are added by comparing the strokes with what they call proxy shapes or ideal shapes like rectangles and circles, and generating a distance between the current shape and the ideal shape. The distance between histogram features is calculated using the earth mover's distance (EMD). Final retrieval is made by generating these features for the input query and then finding indexed strokes with high similarity, the images with high numbers of strokes matched are then retrieved. One major drawback of this approach is that it does not consider spatial relations which can lead to retrieval of many unrelated sketches when applied to math retrieval.

The work by Leung et. Al [36][37] using hierarchical matching is one of the most interesting approaches tried on this field. On this approach, the strokes or primitives are considered to be arranged in certain hierarchy which also needs to be matched before assuming that two sketches are similar. First, the strokes are subdivided into smaller pieces similar to ideal shapes as lines and arcs to create a

multi-form representation of the sketch, one with original strokes, the next one using these subdivisions of strokes, and the last one by merging all the stroke subdivisions that form basic shapes. Then, a graph is built for every representation where each sub-stroke represents a node and each edge represents a relationship between two strokes. The kind of relationship considered is inclusion where one stroke is considered inside another stroke if it is contained on its convex hull. Figure 4 shows an example of the hierarchical representation of the strokes used to draw a simple house with two windows and a door which contains a knob. The spatial relationship between two strokes is described using a displacement vector. Some general features combined with domain-specific features can be used to describe the individual strokes. Two sketches are compared using local and global matching. For local matching, the features of a stroke from one sketch can be compared against the features of a stroke from another sketch. Using this comparison, the distance between all pairs of strokes from the two sketches is obtained and arranged into a matrix. The next step is to find the best matching pairs, and since it is assumed that one stroke from one sketch can corresponding to at most one stroke from the other sketch then the final problem is reduced to the assignment or marriage problem from graph theory. A solution for this problem can be found using the Hungarian method [38], but since this method requires the cost matrix to be squared, then dummy rows or columns have to be added if the numbers of strokes on each sketch are different. There are two possible approaches to consider spatial relationships on this matching process, one is to do it at the time of calculating the similarity between two individual strokes, and the second one is to make it after the assignment has been made as an additional cost above the cost for local matching, and the way it is done in [37] is by adding the spatial matching cost after the assignment has been made. Once that similarity has been calculated between each of the three representations of the sketch, these similarities are combined using weights to yield a final measurement of similarity between the two sketches.

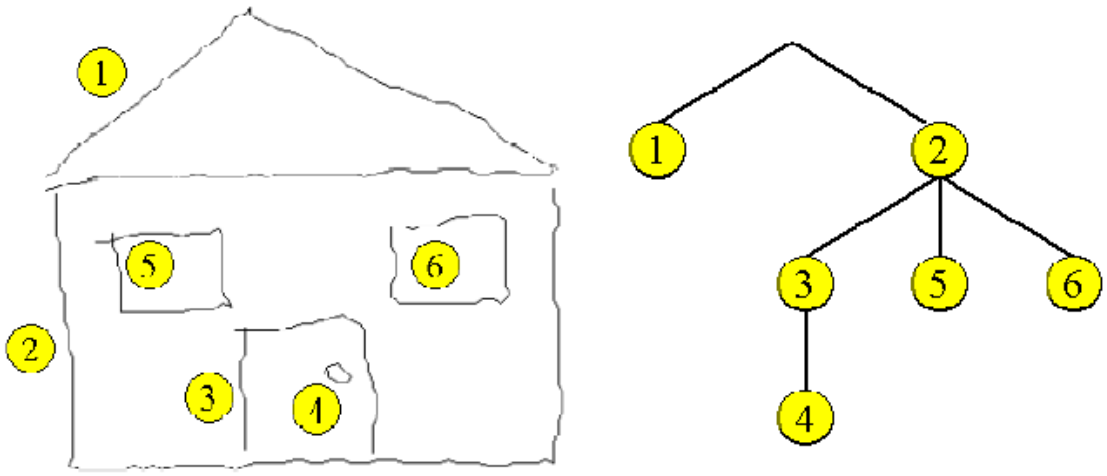


Figure 4. Hierarchical representation of a sketch. Extracted from [37]

This method by Leung seemed like a very interesting option to try on the current system since it is a general method and it also can be adapted to any specific domain just by modifying the selection of features used for local matching. One difference between this approach and the current system is the

fact that the input sketches on the current system are in an off-line format since they will be extracted from images and not from raw stroke data. One of the possible advantages of this method is the fact that it takes the spatial relationships between all pairs of primitives into account, which is desirable for "perfect" matching of formulas with certain arrangement. One foreseen drawback of this method is that making the local matching without considering the spatial relationships first might lead to cases where the estimated similarity is lower than it should. This is due to the fact that constants and variables can appear many times on a single math formula, and there can be cases like the one presented in figure 5. Since the assignment is done using only local matching for visual similarity, and the spatial layout is then compared based on this specific assignment, the system will tell that these two sketches are less similar of what they really are. Another possible disadvantage of this method may be the running time due to all comparisons that have to be performed. Still, this method has the advantage of working with partial matching of sketches which is desirable for the current application. For these reasons, the current system has a partial implementation of this method on its search by recall of matched connected components described in the methodology section.

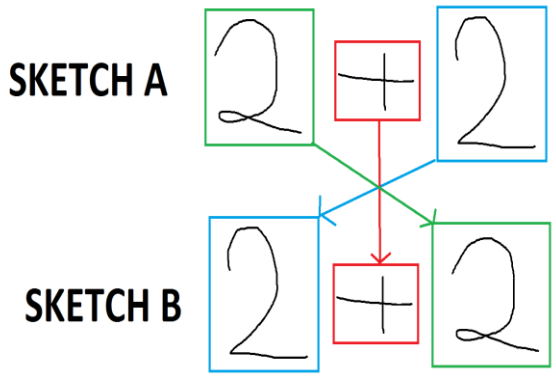


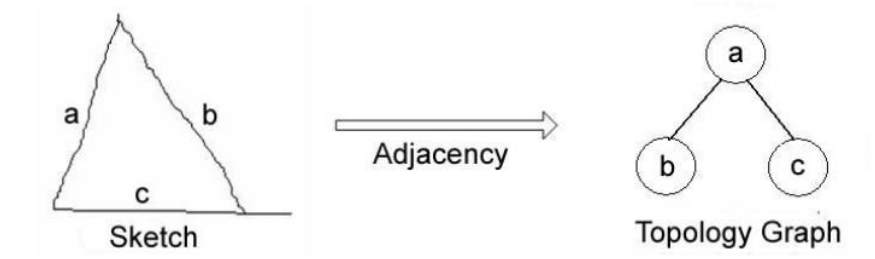
Figure 5. Applying local matching without considering spatial layout first result in cases where two sketches will not be consider as similar as they should.

In general, any measurement of similarity between sketches represented using graphs will require testing several possible combinations and these can grow exponentially as the number of acceptable assignments increases. The general problem of finding the best match that reduces the cost of matching to its minimum both locally and globally can be even worse than the graph isomorphism problem which is known to be an NP problem. However, sometimes approximations of these algorithms can provide feasible solutions for certain applications. An example of these approximations is the test for graph isomorphism proposed in work by Cordella et. al [39] which also has been tested on retrieval of exact matches on technical drawings.

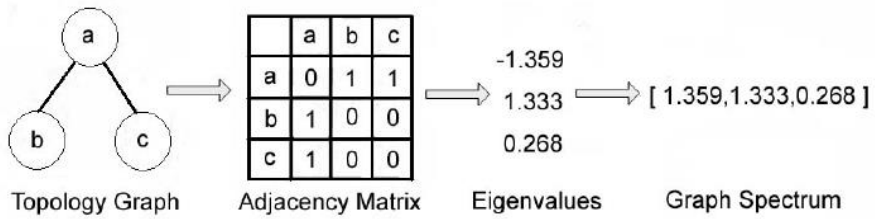
Standard graph isomorphism can determine whether two given graphs are identical or not, but in SBIR the desired goal is to provide a measurement of how similar two graphs are and not only if they are identical. Methods for graph embedding can be used to measure graph similarity and have been applied on many works. The concept of graph embedding consist on creating a vector representation of a graph in a feature space, where a graph represents a point and for any other graph, the more similar it is to the first graph, the closer its points should be to the point that represents the first graph [40]. There are two ways to make graph embedding: implicit and explicit. For the explicit way, an explicit function is

provided which takes one input graph and converts it into a vector representation. For the implicit way, a kernel function is given which takes two graphs as inputs and applies some kind of dot product directly in the graph space producing a final result in vector space. The works by luqman et al. [40][41] represent applications of graph embedding for attributed graphs (graphs with sets of attributes assigned to its nodes and edges) used for retrieval of electronic and architectural diagrams [40][41], letters, fingerprints, molecules for mutagenicity analysis, and general databases of objects [41].

Other methods based on explicit graph embedding have been used before for the field of SBIR. The works by Fonseca et. al [42][43][44][45] describe a method that uses graph spectra to create a vector representation for any given graph. The generation of graph spectra refers to the extraction of the Eigen values of the adjacency matrix [42][45] or Laplacian matrix [43] of a graph and it has been proposed as a really fast approximation of the isomorphism problem. An example of this graph spectra extraction process is shown in Figure 6. The absolute magnitudes of the Eigen values of the adjacency matrices of two graphs tend to be very close for graphs with similar structure. Note that if two graphs have different spectra they are guaranteed not to be isomorphic, but two graphs with the same spectra are not guaranteed to be isomorphic either. In any case, the graph spectra can be thought as some kind of hashing function for graphs with a low collision rate. Also, note that the number of Eigen values depends on the number of columns and rows of the squared matrix from which they calculated, and two graphs with different number of nodes will produce different numbers of Eigen values. However, according to [42] the largest Eigen values tend to be stable for small changes in number of edges and nodes.



(a)



(b)

Figure 6. Getting the graph spectrum of a sketch. Figures extracted from [46]
(a) from the input sketch to a topology graph
(b) from topology graph to graph spectrum

The application of graph spectra to SBIR requires a graph representation similar to the one described by Leung in [36][37] where two vertices (connected components or strokes) are considered adjacent if certain kind of spatial relationship exists. Also, to account for partial matching, the method in [42] describes a multilevel representation which extracts and indexes the graph spectra values measured at different levels in the hierarchy of the graph of a drawing. It is important to mention that graph spectra is used only to match similar topologies between sketches and that additional features have to be used to describe each node for local similarity matching between pairs of components. In the case of [42], geometric features are used to describe each individual strokes.

The method by Fonseca et. al [42][43][44][45] is one of the most interesting approaches since matching topology using graph spectra is faster than methods based on exact graph isomorphism and still faster than methods based on the assignment problem like the work by Leung [36][37]. The method also accounts for partial matching which is required for the current application and domain specific features can be used to match the individual components locally, and therefore a variation of this approach could be used in the future for this project. Given that a set of graphs of sketches from the board have spectra similar to the spectra of an input query, a more sophisticated graph matching procedure can be applied to rank them by detailed similarity.

A variation of the original method proposed by Fonseca et. al [45] is defined in the work by Liang et. al [46][47][48]. Since graph spectra of graphs with different numbers of nodes will be composed by a different number of Eigen values, Liang et. al propose using the norm of the vector formed by the Eigen values as a way to describe the entire topology using a single value. Using this approach, it is expected to have a higher number of collisions of graphs that are similar but not isomorphic. However, one huge advantage of this method is the provision of a really compact, fixed-length representation for the entire topology of a graph. Another important and very interesting difference between this work and the original work by Fonseca is that separated graphs can be used to represent different kinds of hierarchical and spatial relationships between all pairs of elements on a sketch. To be more specific, in [45] eight specific relationships between strokes are used: cross, half-cross, adjacency, parallelism, cut, tangency, embody and ellipse-ellipse intersection. For each graph representation the graph spectra is extracted, and the norm is calculated. At the end of the process, a vector with 8 different values is used to globally describe a sketch, and rough similarity between sketches can be determined using the Euclidean distance between their corresponding vectors. No local-level matching is performed, and no adaptations for partial matching are provided.

Since the work by Liang et. al [46][47][48] is based on the method by Fonseca et. al [45], some of the extensions added in the later works by Fonseca et. al [42][43], like for example partial matching, could be easily adapted on this approach. Also, given the idea that a single sketch can be represented using different graphs, it is possible to adapt the graphs to be used depending on the specific domain where the method is applied. In the case of math retrieval, more specific relationships like super index, sub index, horizontal and vertical neighbors among others, are more adequate to describe the hierarchy of the sketch of a math formula.

In the work by Cao et. al [49] from Microsoft Research, a completely different approach for SBIR is provided. The system creates what they called an “edgel Index” to do fast matching and retrieval of sketches on large databases with millions of images. An “edgel” is a pixel that lies on an edge of an object in an image and it is represented using three values: x-coordinate, y-coordinate, and orientation of the edge. For a given image, the indexation process will first subsample the image to a predefined size of 200x200 and then it will apply an edge detection algorithm generating a list of edgels found. A reversed indexation is done by storing in the index references to the image, one entry for each edgel found as if they were keywords contained on a document. Later retrieval can be done by looking for images containing specific edgels, and ranking of the most similar images to an input sketch is done based on the total number of edgels from the query that were found on each candidate image. The method has been tested on large databases and its really fast retrieving images that generally look like the input sketch, but it has the major drawback of not being translation invariant. With this said, it is evident that this method cannot be used for the current application.

3.7 Measuring similarity between shapes

The measurement of similarity between two given shapes or primitives is really important for the current application. This, however, is one of the most studied problems and several different descriptions have been used and proposed over the years. The same approaches used for SBIR provide their own methods to measure similarity between two given shapes [8][35][36][37][42][43][44][45]. Some of these methods propose using geometric features [42], while other features based on statistics [8]. The number of possibilities is really high, and certain features can yield good results for some specific applications. Since most of the shapes expected on the board represent math symbols and text characters, the final selection will have to be based on different features mainly used previously for OCR [1][50][51]. Only test and error can reveal which of these features will be the strongest for the current application. A very exhaustive list of shape features that could be applied for general shape matching is found in the survey by Yang et. all [52].

3.8 Similar applications

It is important to describe research made that provide solutions for the different problems found in the current application, but it is also important to analyze other similar applications because they can provide additional solutions to the same problems. The application developed in the work by Liwicki and Bunke [1] is a system for automatic indexation of whiteboard notes for a smart meeting room. On this application, three approaches are used: off-line, on-line and hybrid. The kind of input data used in this case is obtained by capturing the strokes as the writer uses the whiteboard. For the off-line approach, the authors convert the on-line data to an off-line format before applying an off-line OCR system. In the proposed project, the data is also obtained partially through a capture device, but no

stroke data is available because the current device used, a MIMIO device, lacks the required API's to capture this raw data directly from the hardware. Another important difference is the use of OCR for indexation of the content of the notes while the current application intends to be recognition-free. However, in their work are identified certain challenges particular to recognition of handwriting on whiteboards that other similar works do not consider at all. One of these challenges is the lack of base lines for text lines in long text paragraphs written on a board, making the segmentation in lines a harder problem than it would be for other handwriting applications. In their work, a very complete algorithm for segmentation of text-lines which can overcome this issue to some extent is provided.

Another similar application that works with videos from lectures is Talkminer [6]. In the case of Talkminer, the input videos are less restricted and contain a variety of shots from lectures from different fields, not only math like in the current application. However, the methods used for indexation rely on OCR of slides present on the video, which means that this content is more similar to printed text than handwritten text. Of course, given the freedom in format of the videos, just identifying slides and text content is already a hard challenge. Another difference is that Talkminer uses keyword indexation which means it will not work for retrieval of math formulas with specific structures unless LaTeX or a similar string representation of math expressions is used.

The Reboard system [7] is specifically focused on indexation and retrieval of whiteboard notes. These notes, however, are not indexed in terms of its content which makes the system less similar to the current application. In the other hand, indexation in Reboard system is made by detecting sets of changes in the whiteboard which is required in the current application too. Only general ideas about the design of such algorithm are present in their paper [7] without including other specific details for its implementation. However, these general ideas were still useful in finding a good approach on the current project.

Thor system [8] is another application that works with indexation and retrieval of whiteboard drawings. This application is one of the most similar in the sense that it can do retrieval of anything written on a whiteboard from off-line data with the difference that it does not consider hierarchy or spatial distribution between strokes, and therefore using the same procedure for math retrieval will result in retrieving math formulas that contain the same elements but not necessarily with the same hierarchy of operations resulting in many non-relevant sketches being retrieved. Also, many similar structures using different operators will not be retrieved, which can lead to many relevant segments not being retrieved. An important contribution of the Thor system [8] is the very detailed description of its preprocessing algorithms which take an input image of the whiteboard and extract stroke data, in other words, on-line data is estimated from off-line data using very sophisticated algorithms which are also provided.

Another related system is the whiteboard capture system developed by Microsoft [9] that works for automatic visual indexation of whiteboard notes on videos from meetings. This system is comparable to the Reboard system [7] in terms of system functionality, and it also does not perform indexation based on the content of the images but based on changes on the whiteboard. In addition, input data for this system comes in the form of video and audio channels, which makes it more related to the current

system. A change detection algorithm from the video of the whiteboard is provided including many details required for its implementation.

Finally, the work by Leung [37] is one of the most related applications in the sense that it was designed to work for retrieval of content written on a whiteboard considering the spatial hierarchy of the elements in the drawing. The kind of drawing that were used to test the system in [37] were Chinese characters and very simple drawings which are very different from math formulas, and therefore applying this approach to a new type of drawing, math formulas, is an interesting research direction.

The presented list of approaches is not exhaustive at all as it only contains a few applications that were considered relevant to the current system because they provide solutions to specific problems found in the current data. It is also important to keep in mind that the proposed methodology intends to be recognition-free and that leaves out many possible approaches while it does not necessarily puts an upper bound to the accuracy that the system could obtain using approaches originally designed for sketch retrieval. The next section discusses the datasets that were used during the development and testing of the final system and will provide a few examples of the input data.

4. Dataset

The dataset used for the developed application consist on a small collection of videos from linear algebra lectures recorded at Rochester Institute of Technology. For each lecture, a still camera has been set in the classroom to record exactly one whiteboard and everything the professor writes on it. As an additional input, each recording comes with an auxiliary video of the strokes of the board captured using a Mimio Capture device and the Mimio software. Figure 7 and Figure 8 illustrate these two video sources. In figure 7 is shown an electronic device attached to the left side of the whiteboard which is the Mimio Capture device mentioned before.

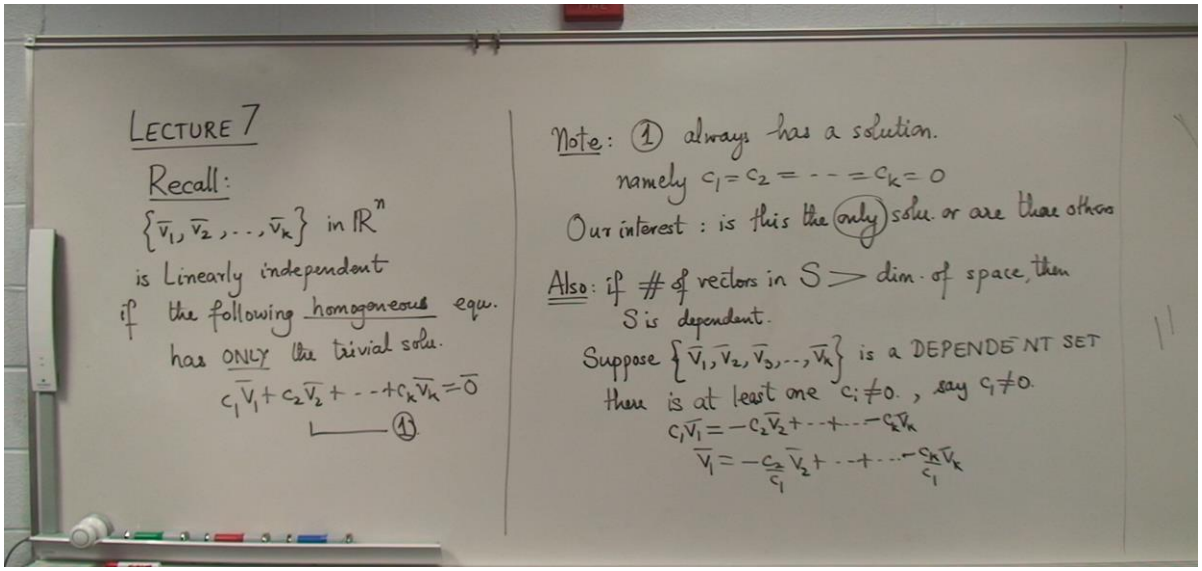


Figure 7. An example of a frame extracted from a video of the still camera in a math lecture

The video coming from the still camera represents the main source that will be used for most of the algorithms to implement for SBIR. An example of a frame extracted from a video of this type is presented on figure 7. Since the goal is to be able to retrieve parts of these videos using the content written on the whiteboard, the quality of the image is critical. For this reason, a full HD camera was used to record each video using a resolution of 1440x1080 pixels. Also, as it was noted in works like the Reboard system [7] and Thor system [8], one typical feature of digital cameras is autofocusing which is completely undesirable for this application because it produces changes on the quality of the traces drawn on the board every time that the speaker moves in front of the board. To avoid this undesired change in quality and to make the input image more stable, the autofocusing feature of the digital camera was turned off before every recording, and manual focusing was used instead. The quality obtained on most of the videos using the current resolution is good enough to expect that a well-developed system will be able to retrieve most of its content. However, there are some drawbacks on these videos, and one of them is that it captures everything between the camera and the whiteboard including all the movements made by the speaker in front of the board, and then identifying specific regions that have been changed is a difficult task, and requires an approach similar to the one applied in

[9]. Another drawback is the changes on lighting as the lecture goes because ambient light is never constant over time as it has been noted on [8].

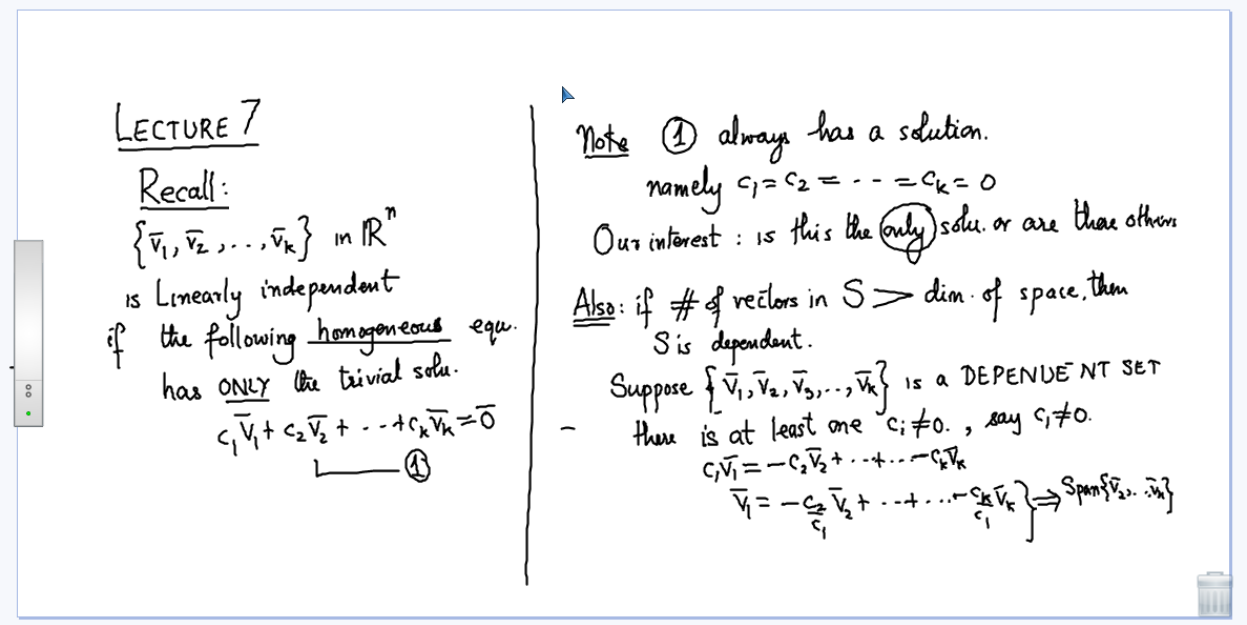


Figure 8. frame extracted from the video of a math lecture as captured by Mimio Software

The video coming from the Mimio software represents an auxiliary source that can be used for purposes that are harder to accomplish on the video from the still camera. The quality of this video is variable as it is basically a screen-captured video and the quality depends on the resolution of the window of the Mimio software. In other words, a computer with a high resolution display plugged to it can capture a video with high resolution too. This video has the great advantage of being cleaner than the video from the still camera in terms of obstructions since it is not affected by the presence of the writer. However, there are some important disadvantages of this video. The first is that the usual quality is not as good as it is required to identify all the strokes as separated connected components on the image. The second is that artifacts can appear from time to time because the Mimio device is very sensible and sometimes just putting the tip on the whiteboard markers makes the system believe that the user is writing, and random points or even small traces are added to the video and stay there for long because they correspond to areas of the board that the writer might have never used in reality. Figure 9 shows an example of these artifacts. The last disadvantage and probably the most important is that Mimio device can miss strokes from time to time producing random incomplete symbols on this source of video, and therefore it is not a reliable source for content. However, since it is cleaner in the sense that the speaker is not recorded within the image, it is a good source for a change detection algorithm that identifies the modified regions and the timestamps of these major changes in content that can be used to extract key frames from the main video. This change detection algorithm has to be aware of the noise present on the video and ignore small random changes.

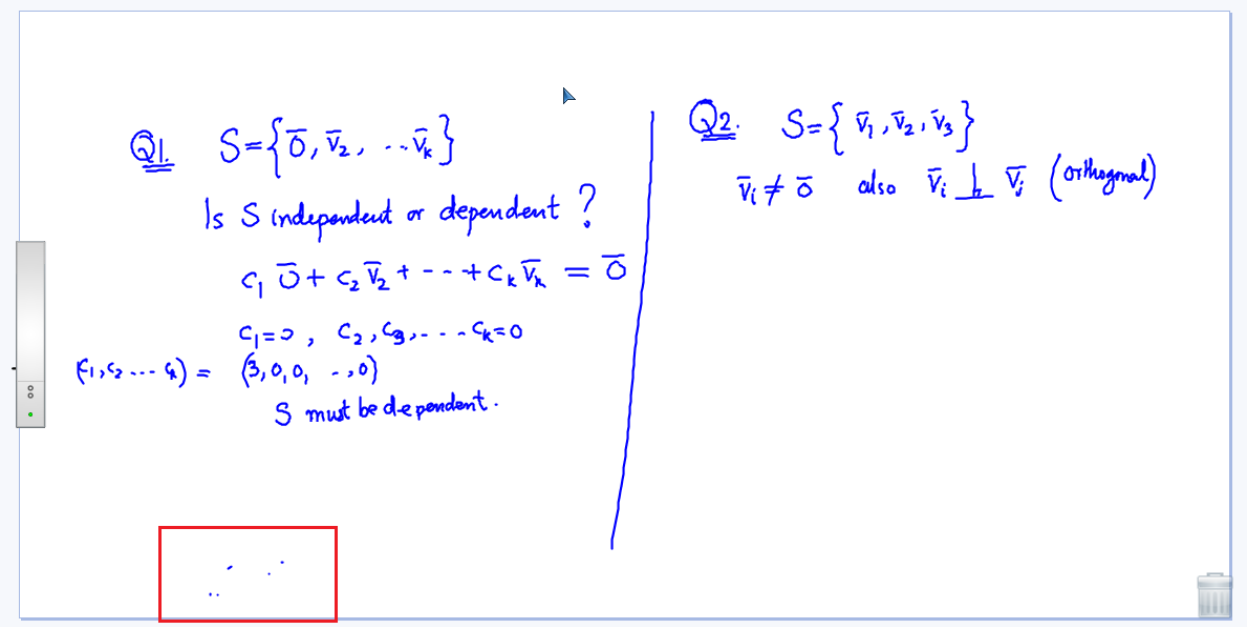


Figure 9. The red box highlights an example of artifacts usually found in Mimio software videos.

The collection of videos used is relatively small in the sense that videos of only 6 lectures were collected. This small quantity of videos is due to the fact that the dataset was still in construction at Rochester Institute of Technology, but in the future larger collections will be available. Also, each video belongs to lectures that last for at least 40 minutes each, making a single video a source of several sketches. Currently 543 sketches were extracted from only 6 videos which mean that 90 sketches in average are extracted from each individual lecture. The input queries are selected by rejection sampling which means that the system selects a random image from the dataset and then the user can accept or reject that image being used as query.

5. Methodology

The process of making available for retrieval the information inside a video of a lecture requires of the combination of different sub processes. Initially, the system starts only with the available videos from the lectures, and by the end of the process, a database of the content in a format that allows its fast retrieval is available. To generate such database, it is required to identify the content on the videos and then add the description of that content to the database. However, just identifying the content inside of the video of a lecture implies many challenges that have to be overcome. The most relevant of these challenges are: synchronization of videos from different sources, visual alignment of content between videos, speaker detection, content change detection, content extraction, and finally content description. Each of these challenges will be described in detail on the following subsections.

5.1 Sketch Extraction

The first part of the process is related to all the work required in order to extract the regions of content from the input videos. For this task, different sub-processes are applied and at the end the system will produce a set of regions of content or sketches that can be further processed for indexation and retrieval. Since the input of the algorithms consists in videos that come from two different sources, certain conciliation tasks must be performed in order to establish an accurate alignment between the content from the two sources in terms of both time and space. Figure 10 shows how the different processes are related to finally end with the extraction of sketches from the main video.

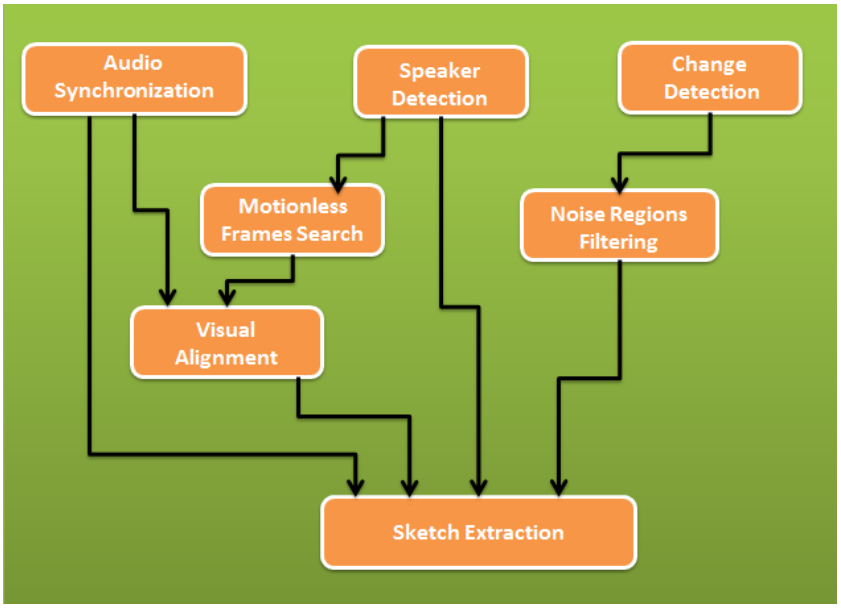


Figure 10. Tasks involved in the first process of converting the input videos into region of content or sketches that will be indexed by later process. Note that certain processes like audio synchronization, speaker detection, and change detection could be executed in parallel as they do not depend on any other task, but the later tasks have dependencies on the results from the previous tasks.

5.1.1 Synchronization of videos

For each lecture in the current dataset, there will be a video that comes from a still camera in the classroom and another that comes from the software of the Mimio capture device. While the two videos are recorded over the same content on the same lecture, these are never guaranteed to start at the same time. Actually, it would be pretty difficult to ensure that both videos will start recording the lecture on the same millisecond or even with less than just three seconds of difference. This is due to the fact that manual operators are required to start each recording and is very difficult to obtain perfect synchronization on this task. Therefore, it is better to assume that these videos are out of synchrony and to apply some method to account for this difference in timing.

One possible option would be to manually edit one of the videos, the one that starts early, and manually remove the additional portion at the beginning to make them start in synchrony. However, it was required to make the entire process as automatized as possible. Actually, the only manual operations applied to input videos are converting them to Windows Media Video format (.wmv) because it is compatible with the OpenCV library for python, and also the extraction of the audio streams as separated files in Wave format (.wav) to process them using the wave library of python.

In order to synchronize two videos of the same content that come from two different sources, it is required to find specific features to describe the timeline of each video, and then use these features to find the best alignment between the two timelines by treating them as sequences. In terms of the content of the video, one could say that by identifying the sequence of times at which the writing and erasing events take place on each video, it could be possible to synchronize the videos by finding the best alignment between these two sequences of events. However, while it is easy to identify writing and erasing events on the video captured by the Mimio device, it is pretty hard to identify the exact times at which the instructor starts and stops writing on the video from the still camera. An easier approach to synchronize two videos is obtained by using the audio streams to find the best alignment between them. As it is show in figure 11, the result of this synchronization is a time offset that will be required to align the timelines of the videos in future processes.

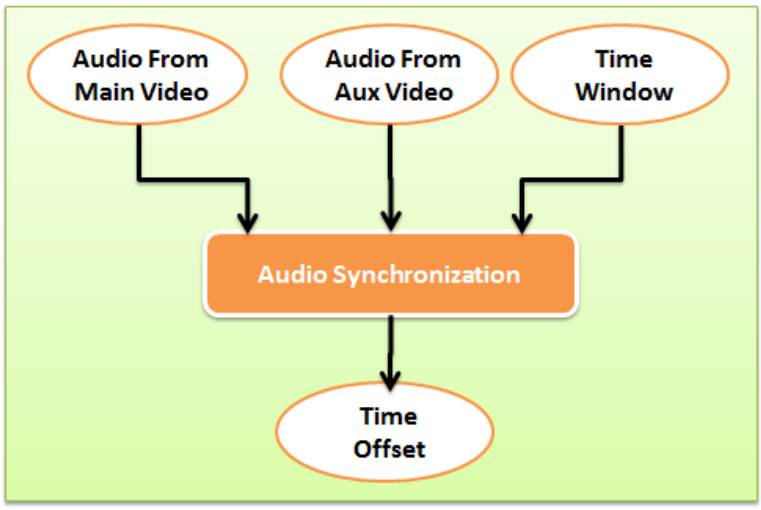


Figure 11. Process of synchronization of videos which is done by aligning their audio streams. An additional parameter that limits the range of the search for the best alignment is required and the result is a time offset that must be applied to the auxiliary video to match the timing in the main video.

To use audio for synchronization, it is important to understand how the digital audio works and then find a way to efficiently match two different sequences of audio. The sound in our world travels through the air in the form of waves. These waves have different frequencies and amplitudes which translate into different sounds and volumes. In digital audio, these waves are stored by sampling the amplitude of the waves at thousands of points inside a single second of audio. Currently the videos are recording using a quality of 44,100 Hz which means that 44,100 points are used to describe the sound of a single second of audio. Figure 12 illustrates the audio waves for just 5 frames from a video.

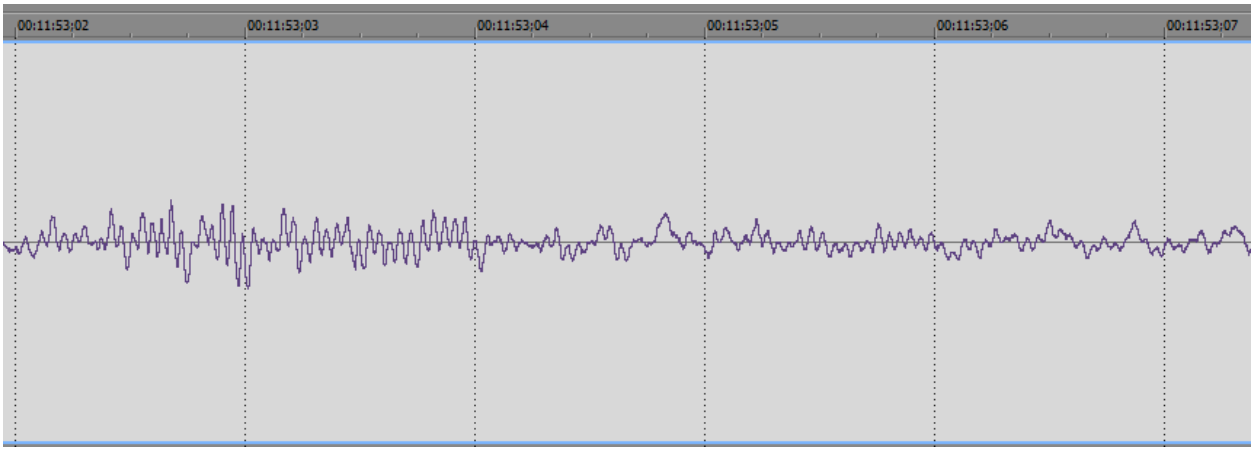
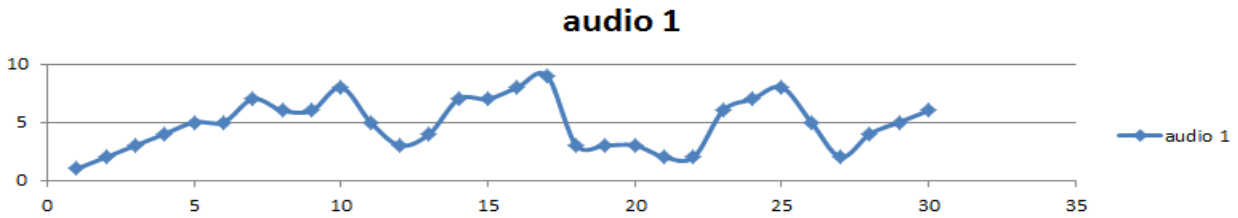


Figure 12. Audio waves for only 5 frames of a video, around 0.16 seconds of audio

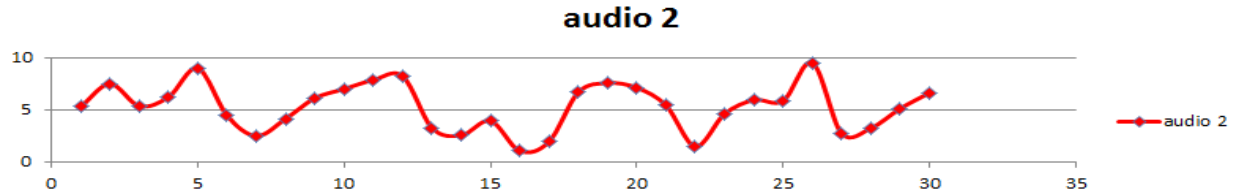
Given that so many points are used to describe a single second of audio, using directly these points to align two videos would be computationally expensive. For this reason, a method for subsampling the audio is required to increase the efficiency of the alignment. Normally subsampling would imply either taking a uniformly selected subset of the samples or obtaining a subset by averaging groups of samples. However in this case none of these would work to describe the sound on a region of time, specially averaging since the waves change sign constantly and the average is likely to become close to zero for most regions. Instead, what is done in the current approach is dividing each second of audio on a fixed number of intervals, in this case 10, and then for each interval obtain the highest peak of amplitude of the audio waves. With these values, is possible to describe one second of audio using only a few peeks, which results in a faster computation of the best alignment for two audio streams with only a small loss of accuracy that can be measured in milliseconds. Another optimization of running time is done by selecting a window of time that limits the maximum expected difference between two videos. In other words, it is known that the videos will start at different times, but the difference is expected to be probably less than a minute. Also, it is not required to use the entire audio stream for synchronization since aligning the sound at the beginning of the video should result in aligned sound at any other part of the video, and for this reason only a small portion of around 8 minutes of sound is currently being used to calculate this alignment.

Even though the two videos represent the same lecture, and are captured on the same place, the audio stream will never be identical even for the exact same portion of time. This is due to the fact that different devices are capturing each stream and the quality of the sound recorded by each of these

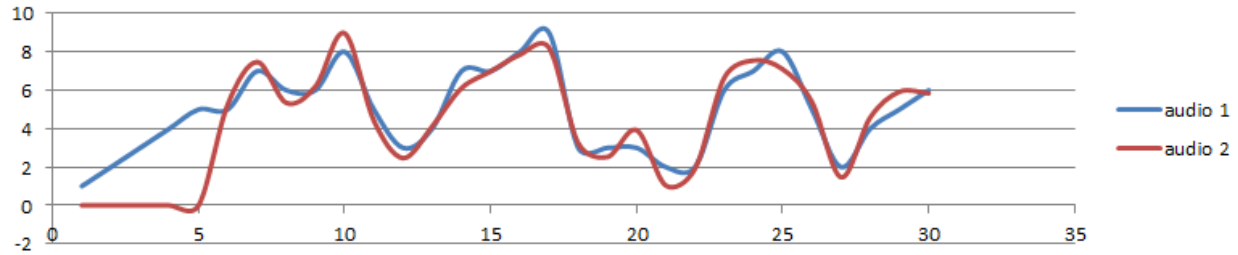
devices will vary as also will vary the amount of noise present on each recording. For this reason, the alignments are calculated by taking the absolute difference between corresponding peaks on a candidate alignment, and adding those absolute differences to estimate a total cost for that alignment. Different alignments are then tested by systematically trying all possible offsets inside of a given window of time, and then the alignment with the lower cost is selected as the best alignment for the two videos. This alignment is described by an offset which corresponds to a certain difference of time between the two videos. This difference will be positive if the second video started recording after the first video or negative if it started recording before. Figure 13 shows an example of the best alignment for two input sequences with a positive offset of 6. Note that while the method is not the most optimal for this task, it still produces acceptable results and the running times in practice can be measured in seconds which is good for a pre-processing step executed only once per lecture.



(a) Input sequence 1



(b) Input sequence 2



(c) Best alignment found for sequences 1 and 2. Current offset is +6.

Figure 13. An example of the best alignment for two given sequences of peaks for two streams of audio

Synchronization between videos is possible only after a correspondence in time has been found. This synchronization will be required for later algorithms, like the sketch extraction algorithm, that work with specific intervals of time marked with time stamps relative to the timeline of one of the videos, then the offset is used to find the same time stamp relative to the other video.

5.1.2 Finding the visual alignment between two videos

The second challenge found in the process of extraction of the whiteboard content from the videos of the lectures was that since the two videos come from different sources, they have the same content on the whiteboard but at different locations. These locations change in terms of absolute pixels at which each character or symbol on the whiteboard can be found on each video, but the relative locations between elements are still the same. In other words, since the two videos have different resolutions and margins for the whiteboard content, it is hard to tell which specific set of pixels from one video correspond to which set pixels on the second video. An algorithm to find such matching was required in order to make possible the mapping of content between videos. This is also known in the literature as the problem of image registration [53].

The main goal of visual alignment of videos is to find a mapping between regions of content of the two videos. This mapping is obtained by applying a transformation to the images of the content on one of the videos to match the content on the second one. For the video captured by software, the whiteboard region is represented by a perfect rectangle. For the video captured by the still camera, the whiteboard region is more like a trapezoidal area due to perspective. In practice, it is still possible to use a rectangular mapping between regions and still achieve acceptable results on some videos. A rectangular mapping can be found by doing a full-search that would test several scaling and translation parameters until a combination that minimizes the difference between the images is obtained. If an image of only the content of the whiteboard is given for each video, it is possible to obtain horizontal and vertical profiles of pixels that describe that content, and using a similar technique to the one applied for video synchronization, an alignment for these pixel profiles can be found. However, after applying this procedure to videos from different lectures it was discovered that the error produced for some of them was too high to be acceptable and that a different technique had to be applied. The technique currently in use is based on extraction of Speeded Up Robust Features (SURF) [54] from different pairs of images to obtain a projection matrix that is used as the final alignment. Also, to ensure a higher accuracy, the SURF are extracted over the images of only the content of the whiteboard on each video.

For the video that comes from a still camera in the classroom, to identify the content of the whiteboard requires some special work since the speaker is present on the image most of the time, and that means that he or she will be obstructing at least one section of that content while present on the image. If it is possible to tell where the speaker is at every frame, then it is also possible to locate frames on the video that contain the whiteboard without obstruction from the speaker. The next subsection describes an algorithm used to estimate the location of the speaker on every frame of the main video, but for now it is just required to know that it is possible to use the output of such algorithm to extract frames from the main video where it is believed to be obstruction-free. Figure 14 contains an example of one frame that was detected as obstruction-free by the speaker detector algorithm. Note that even if the speaker is not present, there are other objects surrounding the board that could interfere with the extraction of the content. Therefore, it is important to remove these objects from the image before generating any feature of the content.

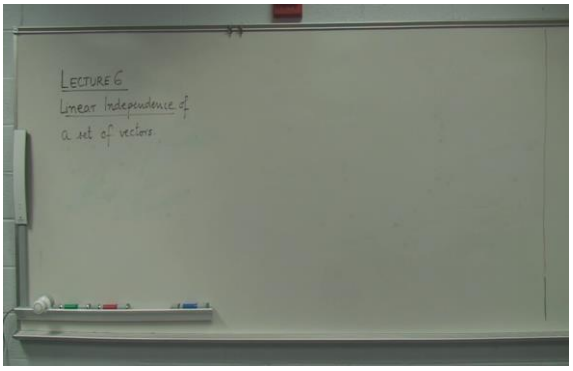


Figure 14. An obstruction-free frame from the still camera video.

Given an image where the content is obstruction free as the example in figure 14, the system can apply a few image operations to extract only the content. Algorithms in image processing usually work with a grayscale version of the image, and the current application is not the exception. With the grayscale version of the image, it is possible to apply a threshold of luminosity to separate dark things like the writing from light things like the whiteboard. However, the whiteboard does not represent a region with uniform brightness. What is more, there will be regions of the board that will be dark enough to be considered writing if just a simple threshold is applied. Also, similarly to dark regions on the board, there will be parts of writing where the traces have a very low level of contrast and these can even represent tones that are lighter than some regions of the board. Figure 15 shows an example of the result of just applying a threshold on a portion of the image in figure 14.

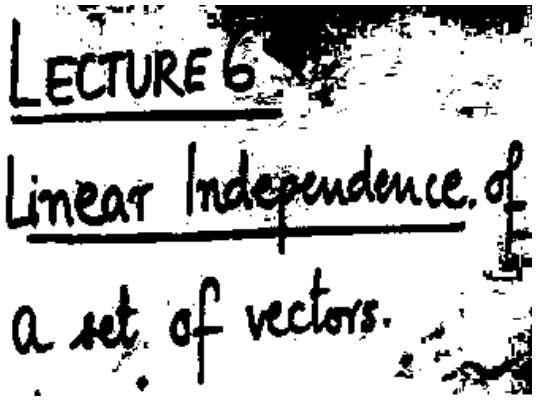
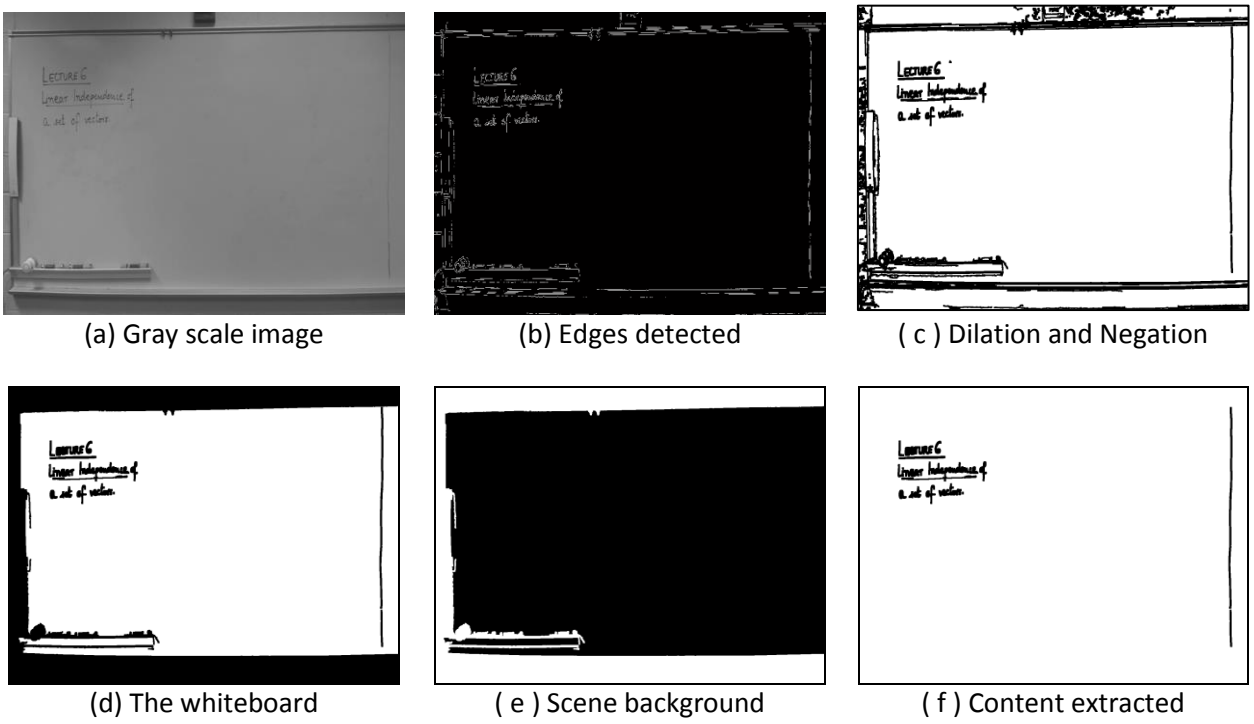


Figure 15. Converting a region of the board to black and white using a threshold without any other pre-processing more than the initial conversion to grayscale.

Since just applying a threshold on the grays-scale image of the video would not help on identifying the exact content on the board, then a different approach is required for this task. It is important to consider that every pixel in the image belongs to exactly one of three possible elements: content, whiteboard and scene background. The current goal is to separate content pixels from the other two kinds of pixels. To achieve this clean separation, the first operation applied is edge detection using the Canny method. Closed regions are needed to separate the edges of the content from other kind of edges on the scene, but the edges obtained with Canny method are not guaranteed to produce

closed regions, and therefore it is required to apply additional morphological operations in order to obtain such closed regions. In this case, a dilation operation using a structural element of 5x5 pixels is applied to close all edges found in the image. After dilation of the edges, the image is then inverted to make large closed region become individual connected components. One example of the results at this point of the process is shown in the image in figure 16.C. Since the whiteboard will usually represent the largest connected component on that image, this largest component will be extracted and threated as whiteboard. The black regions on the image of only the whiteboard connected component can be either part of the content or part of the scene background. Given that regions that belong to the content are usually smaller than background regions, every black region with area above 5% of the total area of the image is considered as background region and is filtered. At the end, the remaining black regions are considered content and used to calculate the SURF for the visual alignment between videos. Figure 16 show the results of each of the steps described before.



(a) Gray scale image (b) Edges detected (c) Dilation and Negation

(d) The whiteboard (e) Scene background (f) Content extracted

Figure 16. Steps applied to extract content of the board for profile alignments

There is no speaker present on the video that comes from the Mimio software, but there are still a few additional objects surrounding the content captured from the whiteboard that must be removed in order to extract only that content. The timestamp on every frame extracted from the main video is used to extract its corresponding frame on the auxiliary video. The same steps are applied to the images extracted from this video and similar results are produced. Figure 17 shows the frame extracted from the auxiliary video that corresponds to the frame shown in figures 14 and 15.

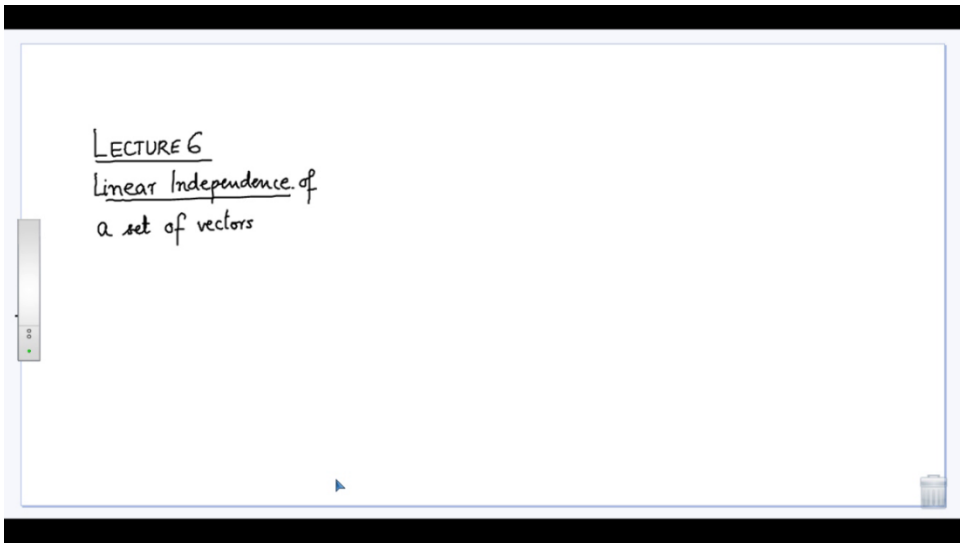


Figure 17. A frame from the auxiliary video that can be used for visual alignment between videos.

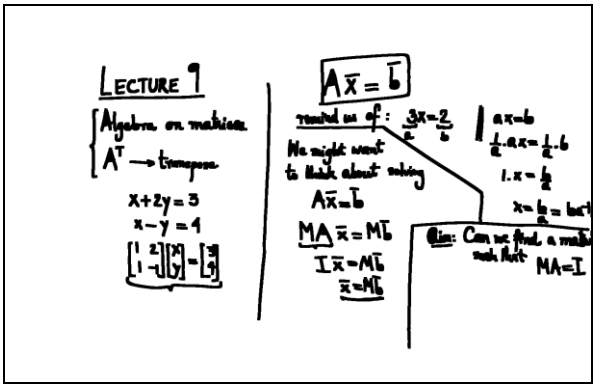
When the content from two corresponding frames has been successfully extracted, the points of interest or key points are calculated on each frame and the SURF are extracted for each of these key points on each frame. The OpenCV library for python contains all the functionality required to easily obtain these features. The next step is to use nearest neighbors to find the best matches between key points of the two images. Since a lot of noise is usually present on the content, especially on the content captured by the Mimio device, the system cannot rely on the matching points of just one pair of frames mainly because on many cases only a small portion of the whiteboard is filled with content. For that reason, different pairs of frames are selected for the process, each of them taken from a different segment of the video. Currently the system uses a total of 25 pairs of frames, and these pairs are selected from different parts of the timeline of the video. After the best matches for a pair of frames have been identified, the next step is to calculate a projection that is consistent with most of these matches, and this is done using the RANSAC method [55]. The way that the method works is by randomly selecting four matching points, getting a projection for them, and then calculating the total of inliers for that projection. The method finally keeps the selection of points with the projection that maximized the total of inliers and returns that projection.

Note that different matching points will produce different projections, and that if 25 pairs of frames are used, then 25 different projections will be obtained. Since the system needs to work with a single final alignment, something needs to be done in order to combine those 25 different results. One option could be to try to average them somehow, but just taking the average of the projection matrices does not work. Another option is to accumulate all the best matching points from the 25 pairs of frames and calculating a single final projection with them. This last approach works to certain extend but the threshold used to select the best matches has a great impact on the results, and finding a particular value that worked for all cases in the current dataset was not possible. Another solution is to rank the 25 alignments and just keep the best one as the final alignment, and this is exactly what the final

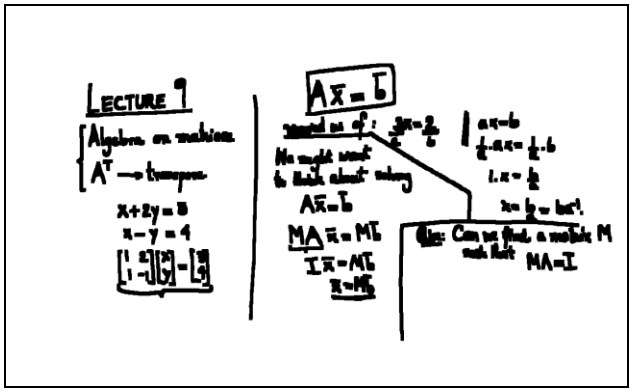
application does. The score for a given projection over a pair of frames can be computed in terms of recall as follows:

$$RECALL = \frac{Total\ of\ content\ pixels\ matched\ by\ projection}{Total\ of\ content\ pixels\ on\ the\ main\ image}$$

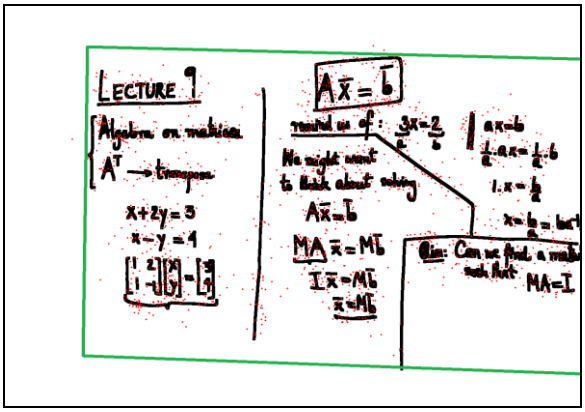
This value is calculated for every projection over every pair of frames, and then the average over all pairs is used to give a final score to each projection. The next step is to pick the projection with the highest average recall as the final projection. This alignment is required for extraction of the content from the main video using the changes detected on the secondary video, and it represents a way to map specific pixels from one video to the other one. Figure 18 shows an example of the projection found between two frames.



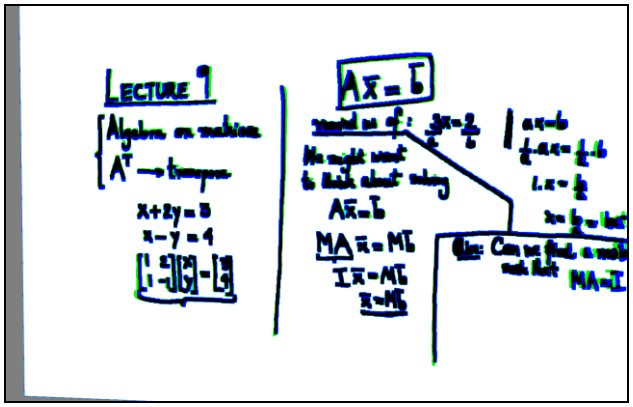
(a) Content extracted from main video



(b) Content Extracted from auxiliary video



(c) Key Points and projection



(d) Final projection of content

Figure 18. Example of the alignment between two frames

a) Image of content extracted from the still camera video (1440x1080 pixels)

b) Image of content extracted from the auxiliary video (1280x720 pixels)

c) Best matches of Key points found used for alignment marked with red dots

d) Final projection. Grayscale of main is using blue channel and projected grayscale of auxiliary is using green channel. Common background is marked white and common content is marked black.

5.1.3 Speaker detection

To do a full detection of the speaker and separate him or her from the rest of the elements on the video of a lecture is a difficult task. However, for the purpose of the current project it is not required to identify where the speaker is at the level of exact pixels. Currently, the system only needs to know a good estimation of the position of the speaker in the video to avoid extracting elements from the content of the whiteboard while the speaker is blocking them. For this reason, the current approach is relatively simple and is based on simple frame differencing to locate individual pixels on the image that have suffered major changes in colors from one frame to another. Large changes in color in one pixel from one frame to another are good indicators of motion found in the video, and no detection of those changes over a region for a long period of time means that the speaker might not be on that region unless he or she is able to stay completely steady for several seconds, which is very unlikely in the current application.

Each single frame on the video of the still camera contains 1440x1080 pixels which is above a million pixels per frame, and also has 29 frames per second with an average length of about 45 minutes. In those terms it can be seen that to process every single frame on the video would be computationally expensive. However, since it is not required to identify the exact pixels where the speaker is present, a subsampling of those frames can help to avoid unnecessary computations. Currently the system only takes 3 frames per second to do its estimations of motion and the results obtained are very reliable. Also, the system does not need to measure the difference between all pixels on two sampled frames because the speaker is large enough in the image to be detected at a very coarse level. For that reason, the system subsamples the frames before calculating the difference, and this subsampling is made by dividing the frame using a grids and then using only one pixel to represent each cell of that grid for the further calculations of motion.

(a) (b)

Figure 19. Some examples of the speaker detector algorithm. The Yellow pixels represent subsampled pixels where motion was detected. The red box represents the area where the speaker is believed to be present based on the center and standard deviation of the pixels with motion. (a) and (b) are two contiguous subsampled frames.

The difference between select pixels is then calculated and if it surpasses certain threshold then motion is assumed on that pixel. The system calculates the average and standard deviation of the coordinates of all pixels where motion was assumed, and uses these parameters to calculate the region where the speaker might be present at that time. Note that because of fast variations of lighting in the room as well because of other factors, there might be pixels where motion was assume when there was none, but since the system takes the average of all those pixels and uses a limited number of standard deviations both in x and y axis, then it is able to discard most of these small variations as simple noise and not as part of the speaker. The speaker position is represented in this case by a box with center equal to the average of coordinates of motion-detected pixels, width equal to 6 standard deviations of the x coordinates, and height equal to 6 standard deviations of the y coordinates of those pixels. Figure 19 shows some results of the current algorithm.

5.1.4 Content change detection

The detection of changes in the content written on the whiteboard is probably one of the most important steps in the current application. Since the speaker will write several things on the board and to do that it is required to erase the content more than once during the entire lecture, the system needs to be able to tell at which time the content appears on the board and at which time it is erased. This content detection could be performed over the main videos, but because the speaker is present on most of the frames the detection of changes on specific areas becomes more complex. What is more, the main video has gradual changes of lighting that could confuse a change detection algorithm. In the other hand, the video that comes from the Mimio software does not have the speaker on it, nor it has variations in the lighting of the board, and for those reasons it was decided to implement the change detection algorithm over this set of videos.

The auxiliary video that comes from the Mimio software currently has a resolution of 1280 x 720 pixels which is still high definition but not as high as the definition of the video that comes from the still camera. The number of pixels present on each frame is still high and to perform the detection of changes on every pair of consecutive frames would unnecessarily expensive. For that reason, the total frames of the video are subsampled to only 3 out of 25 frames per second. The detection of changes using three channels of color is also expensive and for that reason the images are converted to grayscale first and then changes are detected in terms of variations of luminosity of pixels.

If the luminosity of a pixel has a change greater than certain threshold then the pixel is assumed to be changed, either written or erased. While there are not changes of lighting on the video captured by the Mimio software, the software itself seems to try to predict the writing and erasing events as they start taking place. This prediction produces light pixels becoming darker and then lighter again if the writer deviates from the predicted trace. A similar behavior occurs with some pixels when erasing events are detected as they can become lighter but if the direction of erasing changes the software sets them darker again. These small changes can cause confusion on the change detection algorithm by making it believe that erasing events take place while the speaker is writing or vice versa, and for that reason the threshold for changes must be large enough to avoid capturing this kind of noise as real

events, but also small enough to capture real events when the writer uses colors like green that has a very high level of luminosity close to the luminosity of the background.

The changes are detected at the level of individual pixels, but these pixels are grouped into cells that will forms regions of changes. The current size of the cells used is on this implementation is 4x4 pixels. The system first uses simple frame differencing to identify pixels with changes in luminosity. Modified pixels are determined using a threshold over the absolute change in luminosity. If the absolute difference is above the threshold then the pixel is considered a modified pixel. Afterward, the list of modified cells is calculated using the list of modified pixels. For each modified cell the system calculates the minimum luminosity and selects it as the luminosity for the entire cell which means that if at least one single pixel in the cell is dark enough to be considered writing then the entire cell is assumed to be written. There are only two possible events for the cells, and these are: was written and was erased. If a cell becomes written, the system adds it to a region of content. If a cell is erased, different things can happen depending on the state of the content region to which it belongs.

A region of content is a group of written cells and it has four time stamps associated: creation time, last modification time, locked time and erased time. These regions of content are important because they will become the sketches that will be extracted and described by the system. Content regions have three main states: active (or modifiable), locked and erased. An active region is one that has been recently created and is still accepting changes whether these are additions or deletions. Cells can be added to the region through merging, and cells can be deleted from the region through small erasing events. If a modifiable region loses all of its cells because of an erasing event, then the entire region is just deleted permanently. Note that a region of content that has been deleted will not be extracted from the video because it is considered that nothing important could be written and erased on such short lapse of time.

A locked region is a region of content that stopped accepting changes after that a certain lapse of time has passed without being modified. The system locks the regions after a threshold of time when a new change comes that could overwrite or erase the content present on that region. When a region is locked, the system releases all cells that were part of that region and these cells now can become part of new regions. When erasing events are detected over cells that previously belonged to a region currently in locked state, the system registers that time as erasing time for that region and the region passes to the erased state.

The content regions have boxes that define a merging area where all the newly written cells found on this area will become part of the content region. This merging area is illustrated on figure 20. The area is sensible to the expected order of writing by being larger at the right side. When a cell is written and there are no active regions or if it does not fall into the merging area of any active region then the system will create a new region of content for that cell. If a newly written cell falls into the merging area of two or more active content regions then all these regions will be merged into a single larger content region. However, there are a few exceptions to this rule in order to restrict the horizontal growing of these regions to avoid entire columns of content becoming a single content region, and also

to prevent content regions to be merged with special regions like vertical lines drawn by the speaker to separate the content on the board into sections.

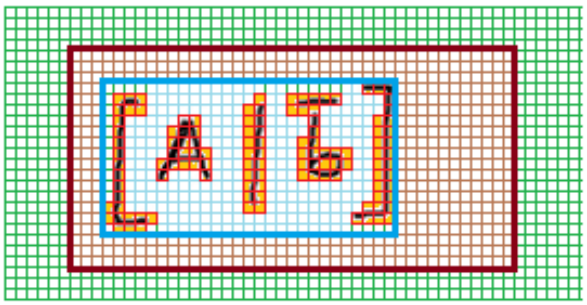


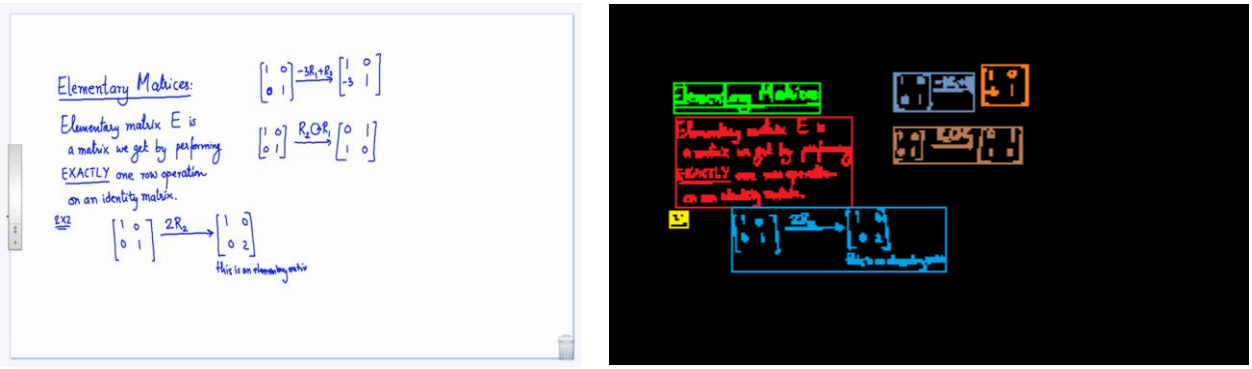
Figure 20. Margins of a content region. On this figure, the orange cells are the cells detected as written on a content region while the blue cells are the non-written cells that form part of the same content region. The brown area is the merging area where if any change is detected then it will be merged with the content region and the green cells are the ones out of the range of the content region

The life cycle of a content region is important because the system will use the detected times for creation, last modification, locking and erasing to groups this regions into key frames. Also, the system needs these time stamps to find frames on the video where the content region is free of obstructions in order to extract them for indexation.

Note that while real content will be grouped into regions that will make it easy to handle, noise is also likely to create its owns regions which have to be detected and removed to keep the index as clean as possible. There are certain properties of the content regions that can be used to separate real content regions from noisy regions. The first property to look at is the size in terms of number of cells used. Real content regions have a certain average size and most regions below 20 cells of content are just noise. The second property is the edition time which is equal to the difference between creation time and last modification time. The edition time is at least 10,000 milliseconds for most of the important content regions. However, there are important regions of content that can be written in less time. For the current application, all regions with edition time lower than 400 milliseconds are considered noise. Density is the third factor used to filter regions, and it is calculated dividing the total number of written cells by the total number of cells on the region. Most of the real content regions will have at least 10% of density of written cells. Finally, the aspect ratio is very important as content regions are neither too long nor too wide. Currently, anything that has an aspect ratio above 10.0 or below 1/10.0 is considered a divisor line on the board and removed from the list of content regions. Of course, there must be exceptions to some of these rules as there are regions that can fail on one or more rules but fulfill the others too well to be considered noise. One example could be a region with size of less than 20 cells but edition time of 5000 milliseconds, such region could not be considered noise because even though it is very small, it took too long to be edited as to be considered random noise.

It is important to note that the current algorithm is far from perfect. Some parts of the content are split into more regions than they should and sometimes the erasing times are registered before the real erasing times making the system believe that content regions have shorter life spans than what they actually do. However, results are still acceptable in most of the cases and the resulting regions are very useful divisions of content that make the whole indexation process a lot easier to handle. An additional algorithm to group these content regions into key frames base on their time stamps is applied during

the indexation process described on section 5.2. Figure 21 shows an example of the results of the change detection algorithm over a period of time.



(a) (b)

Figure 21. Results of the change detection algorithm: (a) An image from a certain segment of a video, (b) The corresponding regions formed by the change detection algorithm shown with different colors

5.1.5 Extraction of whiteboard content

The extraction of the content is probably one of the most important elements on this project since sketch retrieval will be done over the content detected and extracted from the videos. Basically, all the tasks described before are just auxiliary tasks required in order to make extraction of content possible. The content on the whiteboard is present on both the main and the auxiliary videos, and while the auxiliary video has the great advantage of having the content free of obstructions, it also has the great disadvantage of being really noisy in the practice. Figure 22 shows an example of how noisy the auxiliary video can become due to sensor errors among other problems. For the reason mentioned before, the extraction of the content is done over the main video instead of using the auxiliary videos. The mapping of content between the two videos is mainly required because the process of change detection and the process of content extraction are executed over different sets of videos.

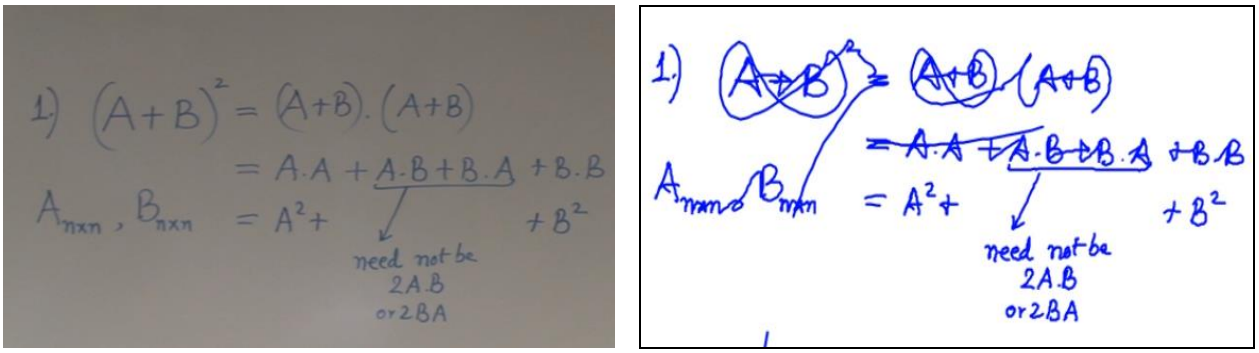


Figure 22. Presence of really noisy data on the Mimio video. On the left, part of a frame from the main video On the right, the parallel frame from the auxiliary video as captured by the Mimio device.

Different elements are involved in the process of sketch extraction. Figure 23 shows the minimum input required for the sketch extraction algorithm. First of all, the algorithm needs to receive the list of the names of the files that compose the main video considering that the camera automatically splits the video every time the current recording reaches 2GB of size. The next element is the time offset calculated to synchronize the videos because the time stamps associated to each regions of content are relative to the auxiliary video, and to find the corresponding frames on the main video the system needs the synchronization time offset. The filtered regions of content represent what the algorithm must extract from the main video and the motion detected on the main video is required to find frames where these regions are not being blocked by the speaker. Finally, because the regions of content have boundaries relative to the auxiliary video, the visual alignment is needed to make the extraction of their corresponding pixels on the main video. The boundaries of the regions of content are projected over the main video to extract the box that contains the entire region.

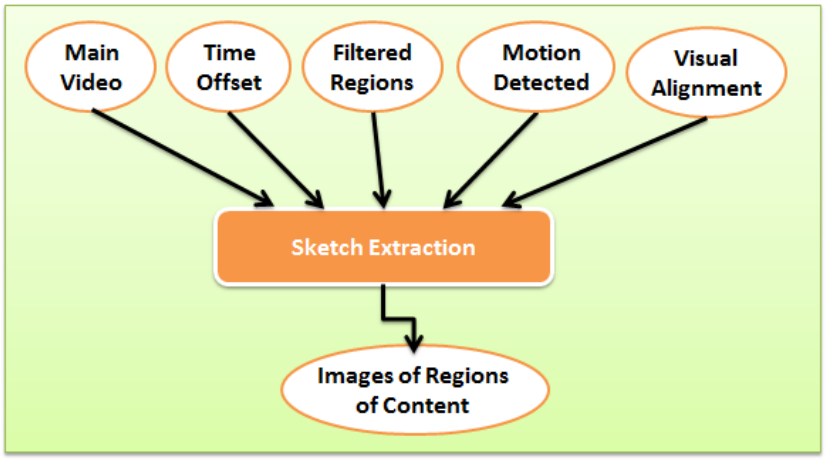


Figure 23. The inputs and the output of the Sketch extraction algorithm.

Note that while changes are detected at the level of cells, the system will extract the final sketches as rectangular images. What is more, these estimated rectangles are expanded by some small margin just to count for small errors in the projection of the visual alignment to avoid cutting connected components. In some cases the final sketches can contain parts of content that were inside of that rectangular region but were not part of the original cells that represented the region. These parts of the content can be considered as noise and they are also hard to remove in most of the cases. Figure 24 shows an example of this kind of noise.

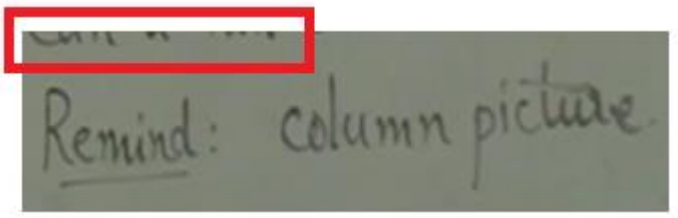
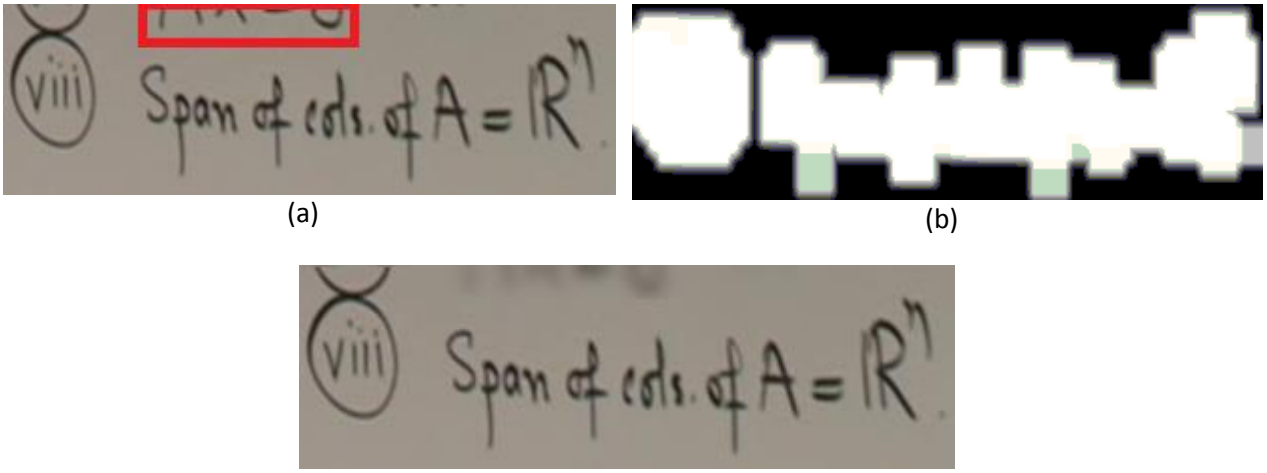


Figure 24. An example where parts of other sketches are accidentally extracted because they fall inside the projected rectangle of the current region. The red rectangle marks the location of this noise.

One procedure tested for removal of this noise was to generate a mask using the cells that represented the region on the Mimio video and then applying a projected version of that mask over the main video during the extraction, but this led to other problems that needed to be solved. First of all, the projection is not guaranteed to fit perfectly the content, and for this reason some dilation operations were applied to the projected mask to ensure that it would cover all the content pixels. The next problem is that the mask only defines which pixels should be extracted, but then something must be placed instead of the pixels that were filtered by the mask, and that filler is important because the wrong filler will generate false edges on the extracted image. White or black colors are examples of wrong fillers because their contrast with the average color of the whiteboard is too large and generates edges. Then, another option is to replace those pixels with the color of the whiteboard that is not constant but can be estimated for each region by averaging the color of the pixels that were filtered out. This works in most of cases but still generates edges especially on large sketches. The last option tested was to blur a copy of the whole image and use the fuzzy version as the filler for the filtered pixels. This blurred version limits the edge detection on additional components contained on the rectangle. Figure 25 shows some results of this procedure.



(a) (b)

(c)

Figure 25. Successful removal of noise.
a) Original input region, noise to remove marked by red rectangle
b) Mask generated by projecting the cells of the content region over the whiteboard
c) Final result by replacing the black pixels on the mask with a blurred version of the image.

While the algorithm described before represents a potential solution to the problem of additional connected components, at the end it was decided to keep those additional components as part of the content as there are also many cases where Mimio misses strokes of the writing on the whiteboard, and then applying the previous method makes the system to miss those strokes on the main video too. Figure 26 illustrate one case of many missed strokes over a single region. Note that in general terms, it can be said that it is worst to lose real content than to keep noisy content.

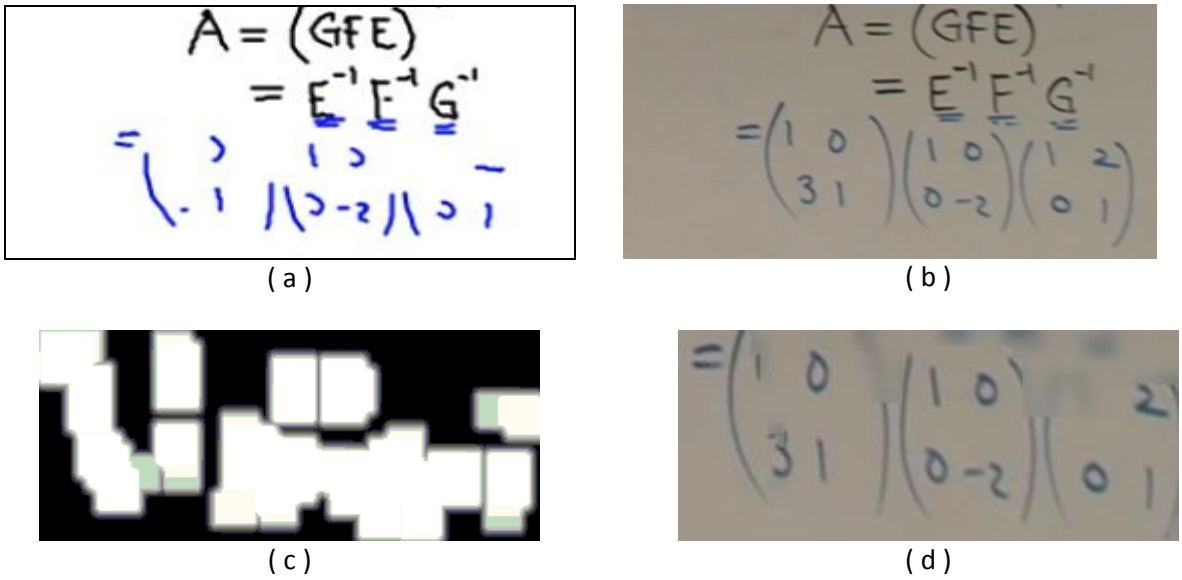


Figure 26. Missed strokes by the Mimio device.
a) The whiteboard as captured by the Mimio device. Many strokes went missing.
b) The real state of the whiteboard.
c) The projected mask created by the algorithm of noise elimination.
d) The resulting region with real content erased as noise.

Many of the extracted regions will therefore contain some noisy connected components that still could be detected and erased by a more sophisticated method, but for now that belongs to future work. Once that all the regions have been extracted and their time stamps are known, the system is ready for indexation of content. The next section describes the process of sketch indexing and all the challenges faced during the development of this second process of the system.

5.2 Sketch Indexing

After the different regions of content have been identified and extracted from the videos, the system must describe the content found on those regions and store those pre-computed descriptions in a format that will make fast search possible. The indexing procedure is directly related to retrieval method since all information needed for retrieval must be pre-computed at indexing time to avoid unnecessary repeated calculations at search time. The current implementation stores the individual connected components found on each sketch, and also their corresponding features. Then, the system also generates estimated key frames based on the time stamps stored along with each sketch, and these key frames are stored on the index for later searches by key frame. Finally, the system also stores neighbor graphs that are generated over the sketches and key frames to describe their structure for partial structural matching. The following subsections describe each of these processes in detail.

5.2.1 Extraction of connected components

Connected components represent the basic units of content on the current application and their extraction must be as accurate as possible. Ideally, a connected component should be created for each individual symbol but this is hard to accomplish on many cases. Also, the input images represent handwriting and compared to images of printed text the level of variation is much higher. For example, the speaker can use print and cursive writing interchangeably as shown in figure 27, the result is a mix where many symbols are written using a single trace and many traces represent a single symbol. The case where a single symbol is written with many traces does not represent a problem as long as the writer is consistent with that writing style. The real problem is when two or more symbols are written using a single trace or if the traces of two or more symbols are touching traces because separating them into individual connected components is a hard problem out of the current scope of this project.

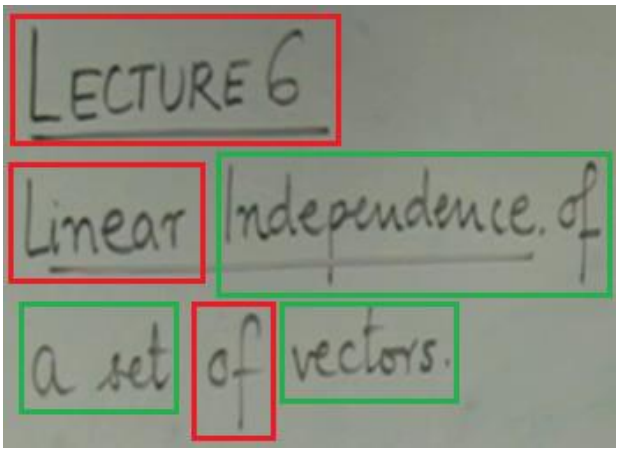


Figure 27. A single region of content with high variations in writing style. The red rectangles enclose print characters while the green rectangles enclose cursive writing. Note that the word “of” appears twice on this region written in two different ways.

Different image processing techniques must be applied in order to extract the connected components of the content written on the whiteboard. Each of these techniques is error prone and the resulting connected components are not always representations of individual symbols written on the board especially because of many touching symbols present on the board. However, the results for most of the math formulas are good enough to allow them to be retrieved even by partial matching with queries. In general terms, the algorithm to extract the connected components is as follows: Find the potential content, remove the background, classify remaining pixels as either whiteboard or content, and then use connected components labeling to find the resulting components.

To find the potential content the system uses edge detection to locate pixels that surrounding the writing. The edges are detected using Canny edge detection algorithm. Note that it could be possible to apply first a threshold over the grayscale image but the results would be similar to the problem described on content extraction for visual alignment of the videos. There are pixels of writing with higher luminosity than some pixels of the whiteboard, and for that reason edges detection is required to detect potential content pixels based on the local contrast of each pixel instead of using a single threshold of luminosity.

The content pixels are enclosed by the edges detected but these edges usually do not form completely closed regions. A dilation operation is then applied to close the regions found by the edges, but note that this operation not only will close the content pixels, it will also expand the edge pixels toward the whiteboard pixels. At this point most of the real content pixels should be inside the expanded edges, and also the large black regions should be part of the whiteboard. The next step is to get the negated version of the dilated edges image, and then to label the connected components on that image. For each connected component, if it is above certain threshold then it is considered a whiteboard region. After all whiteboard regions have been identified, they are merged to create a single mask which is dilated as much as the edges were dilated before. The result is a mask that contains only pixels that can be safely assumed as whiteboard pixels. The main reason of why a threshold is applied before adding a region to the mask is because sometimes holes inside of the characters get expanded too much and end up with including content pixels as part of the whiteboard. All the steps described before produce the partial result shown in figure 28.e.

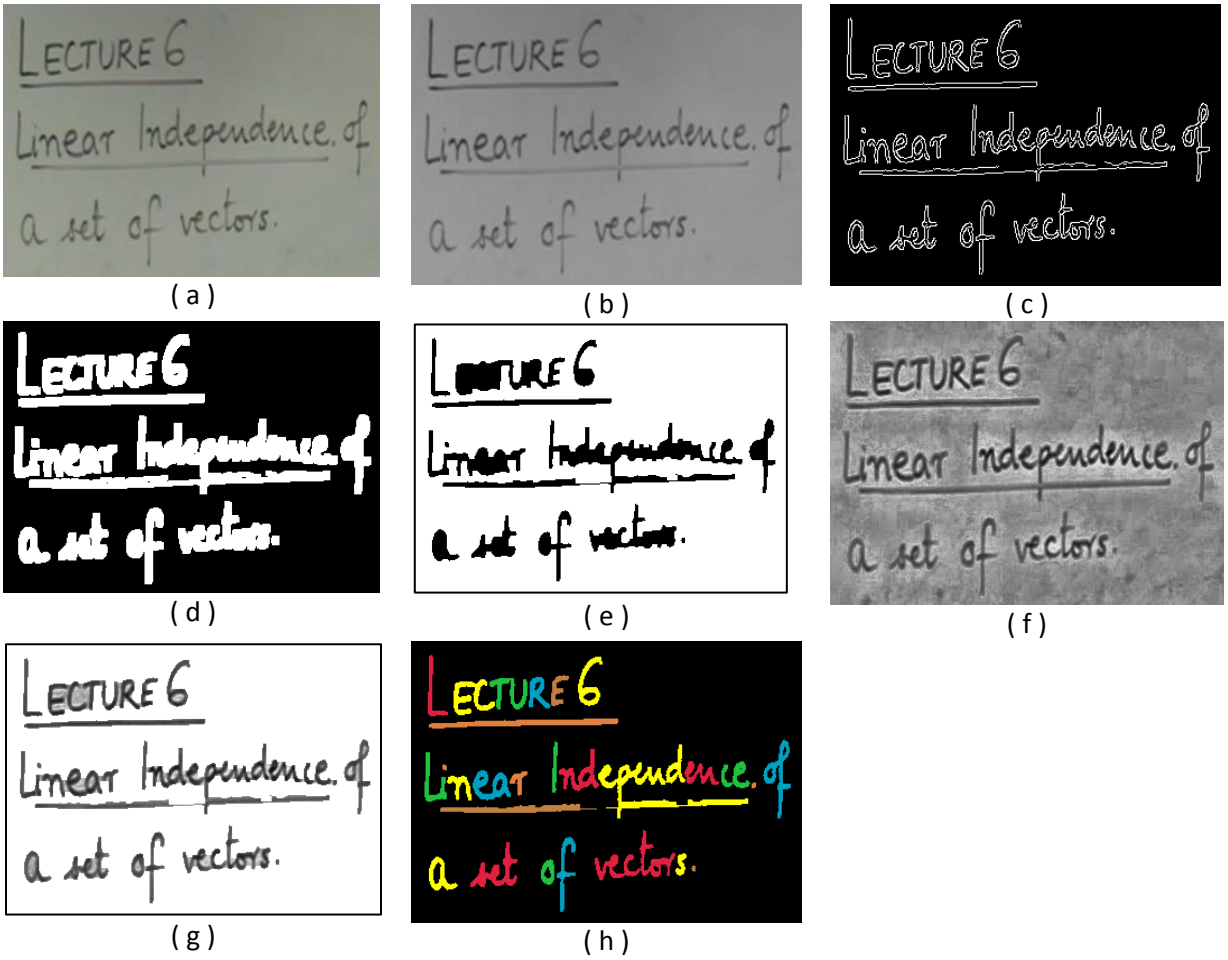


Figure 28. Steps to extract connected components. (a) input image, (b) converted to grayscale, (c) Edge Detection, (d) Dilation of edges, (e) Whiteboard Mask, (f) Adaptive Histogram Equalization, (g) Applying the mask (e) to equalized (f), (h) final connected components after labeling.

Most of the pixels of the whiteboard should be correctly identified at this part of the process, but now it is required to separate the remaining whiteboard pixels from the content pixels. It is possible to apply a threshold over the remaining pixels and the noise present is for sure smaller than when the process began. However, the problem of certain parts of the content with lower luminosity than certain parts of the board remains. A way to overcome this problem is by applying adaptive histogram equalization on a copy of the original image to enhance the local contrast between the writing and the whiteboard, and then apply the threshold over the image with enhanced contrast. Note that using mask of the whiteboard becomes mandatory when using adaptive histogram equalization since it will make some regions of the whiteboard really darker specially when there are no dark pixels of content on the region. The equalized image will have a darker background than the original, but the background will become lighter on the pixels surrounding the writing. After the image is equalized, the whiteboard mask is applied, and then a threshold is used to separate the final pixels of content. The last step is to find the connected components on the resulting image, and this will be the final connected component used by the system. Figure 28 illustrates the entire process, from an input sketch to the extraction of the individual connect component.

5.2.2 Describing the connected components

The next step after the connected components have been successfully extracted is to generate their descriptions. This is done using different features and combining all of them into a single feature vector. The features to use are really important as bad features result in low accuracy when trying to find similar elements on the index. The current set of features is very simple and in practice achieves reasonable results. The features used are based on features applied for offline optical character recognition, and while the current system does not apply classification to identify the individual symbols, it needs to be able to tell how similar two symbols are.

The only preprocessing applied to the characters before the extraction of its features is to normalize their images to a standardized size of 128x128 pixels. If the input image has a size of less than 12x12 pixels then the character is padded with black pixels to make the input image have a size of at least 12x12 pixels. This padding is done with the purpose of avoiding some extra small components being expanded more than 10 times their original sizes. Also, if the input image is not squared, then it is padded with black pixels to make it squared to avoid changing the aspect ratio at the time of resizing. After the image has been resized, the system extracts the following features: aspect ratio (1 value), center of mass (2 values), covariance matrix (3 values), crossings (16x3x2 = 96 values), and 2D histograms (5x5 = 25 values). The current length of the feature vector is 127 values.

General Features

These features correspond to general values that are calculated over all the white pixels of the normalized image. These features are: aspect ratio, center of mass, and covariance matrix. The first feature is the aspect ratio of the bounding box of white pixels which could be obtained by dividing the width by the height of that box. Horizontal and vertical lines are the most extreme values for this

feature while squared components should be in the middle of the scale. However, if width is always the numerator and height is always the denominator, the system will fail to measure the real similarity between pairs of components. For example, if two elements are given, one of them is 20x10 and the second is 30x10, their aspect ratios would be 2.0 and 3.0 respectively, and the difference would be 3.0 – 2.0 = 1.0. In the other hand, if the same elements are rotated 90 degrees, the first one would become 10x20 and the second 10x30, with aspect ratios of 0.5 and 0.333… respectively, and the new difference would be just 0.5 – 0.333… = 0.1667… which means that the same two components would be considered more similar by just rotating them, and that is completely wrong. A best way to get the aspect ratio is by swapping the numerator and denominator by using always the largest one as numerator and the smaller one as numerator. As a result, the value will be always greater than or equal to 1.0. Of course, a difference must be made to make 20x10 different from 10x20, and this can be done by swapping the sign of the aspect ratio when the denominator changes. This change creates a discontinuity on the function when the aspect ratio approximates -1.0 and then it changes to 1.0. To solve the discontinuity, 1.0 can be subtracted from the value of the division before swapping the sign, and it results on a continuous function equal to 0 when the input size is a perfect square, negative when the input is longer than wider and positive otherwise. Finally, some scaling factor is applied to control the influence of this value on the final distance. The final formula is as follows:

$$Aspect\ Ratio = \begin{cases} \left(\left(\frac{width}{height}\right) - 1.0\right) x\ scale\ \ if\ width \geq height \\ -\left(\left(\frac{height}{width}\right) - 1.0\right) x\ scale\ if\ width < height \end{cases}$$

The second general feature is the center of mass which is simply getting the average of all x and y coordinates of all white pixels P on the normalized image. If the raw average is used the resulting scale would be too large and it would heavily affect the measurement of distance. For that reason, the value is scaled and translated in a way that it will be in the range between -1 to 1 for each axis. The final formula is as follows:

$$H = \frac{Normalized\ Size}{2.0}$$

$$Center\ of\ Mass = \left\langle \frac{\sum_{i=0}^{N-1} \frac{(P_{ix} - H)}{H}}{N} \middle| \frac{\sum_{i=0}^{N-1} \frac{(P_{iy} - H)}{H}}{N} \right\rangle$$

The last general feature is the covariance matrix of the coordinates of pixels. Basically, these values would roughly describe the distribution of the white pixels on the image. Note that the covariance matrix is symmetric, and therefore if there are only two dimensions then only three different values would be needed to describe the matrix. The first value is the covariance (x, x) which is also the variance of the x coordinates, the second value is the covariance (x, y) and the last value is the covariance (y, y) which is the variance of the y coordinates. The formula is as follows:

$$H = \frac{Normalized\ Size}{2.0}$$

$$Covariance(A,B) = \frac{\sum_{i=0}^{N-1}\left(\left(\frac{(P_{iA}-H)}{H} - Center\ of\ Mass_A\right) x \left(\frac{(P_{iB}-H)}{H} - Center\ of\ Mass_B\right)\right)}{N}$$

Crossings

This is a very simple but effective feature used to describe shapes of the connected components. The general idea is to use lines at certain predefined positions and then for each line count the number of times that the line crosses the connected component. Also, additional values can be extracted that help to produce a more complete description of the component at that position and these values are the relative position of the first and the last time that the line crossed the connected component. The system currently uses 16 horizontal lines and 16 vertical lines for a total of 32 lines generating 3 values each. In total, 96 values are generated by the crossings features. The scaled images usually make the connected components very thick and for that reason the system will use always the center of the interval of white pixels crossed as the position of the crossing. One special case arises when a line has 0 crossings with the connected component since the first and last crossings are technically undefined. In that case, the system creates an assignment that cannot be obtained in normal circumstances by making first and last equal to the highest and lowest possible values respectively. Normally these two values would be equal if the count of crossings is equal to 1, but position of last would never be smaller than position of first. Figure 29 shows an illustration of the crossing features.

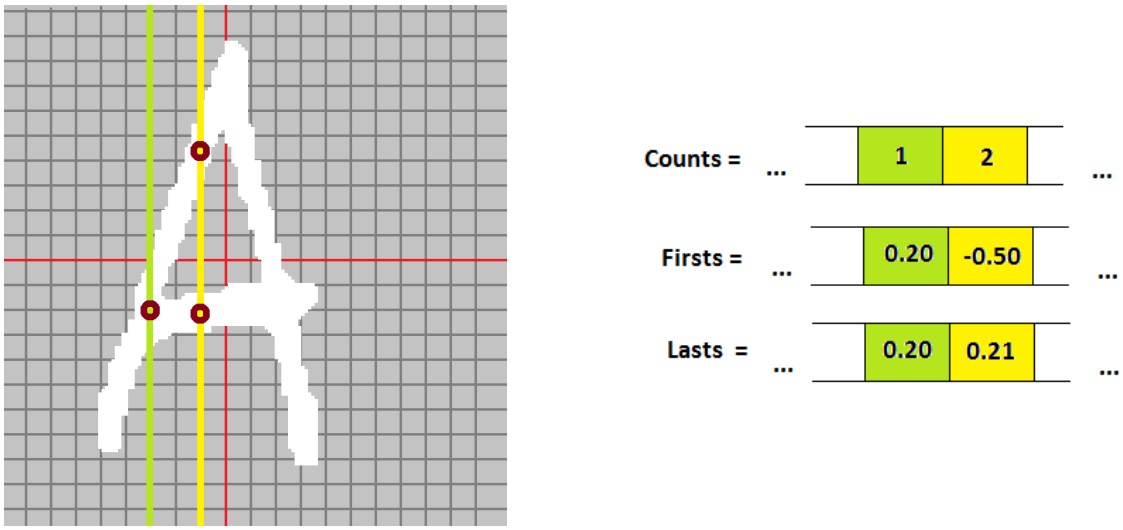


Figure 29. Examples of two vertical crossings to describe an A at fixed x coordinates. The Green line is the first crossing, and the yellow line is the second crossing. The red axes represent the center of the normalized image. The dark red circles represent the center of the crossing interval at which the position is extracted.

Note that if the raw values of counts are used in the feature vector then the counts would become heavily weighted over other features. Then, a normalization process is applied to get the values of the counts on a smaller range currently from -3 to 3 where 0 crossings becomes -3 and 10 or more crossings becomes 3.

2D Histograms

The last feature used is the 2D histogram of pixels which describes the distribution of the white pixels on the normalized images. The system divides the normalized image into a grid of 5 x 5 cells, and then for each cell it counts the total number of white pixels found on that cell. Then, the total of white pixels is used to normalize the values of each cell. At this point the sum of all cells is equal to 1.0. In this case the average value for each cell is on a very small range because most of the cells will be equal to 0.0 and usually the others are never above 0.2. It was tested empirically that the contribution made by this vector to the measurement of distance is not significant unless the values are scaled to make them larger. It was found that a scale of 10 times the original value makes them have an important weight on the calculation of similarity distances between connected components. Figure 30 illustrates this feature for a connected component representing the A symbol.

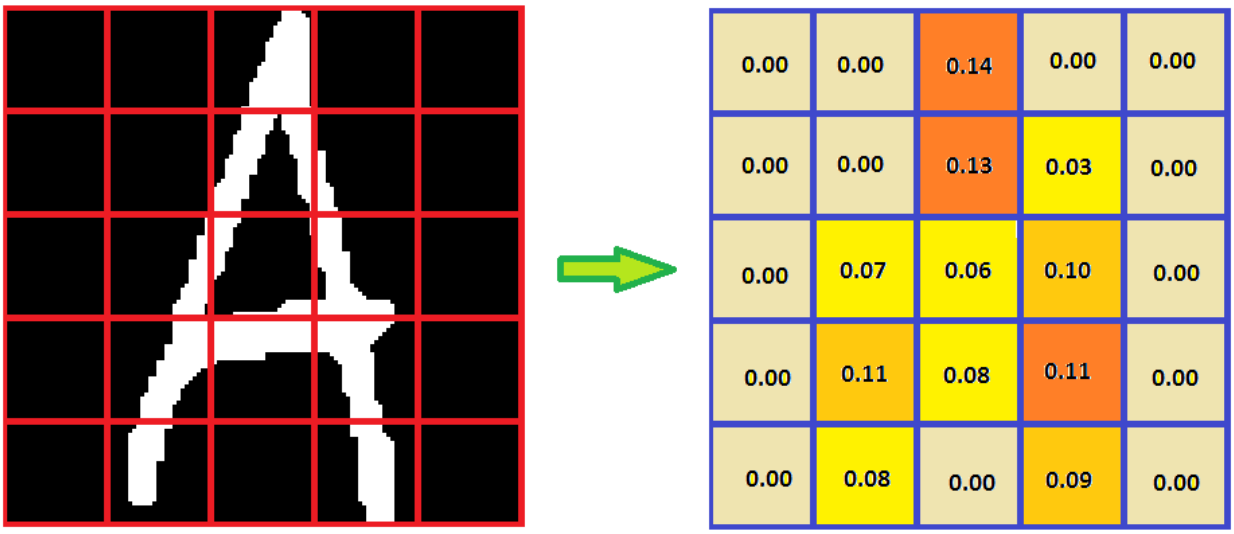


Figure 30. 2D Histogram feature example for the letter A.

When all the features described before have been computed for each connected component, the system can store them and just load the pre-computed values at the time of retrieval. The time required to compute these features for a single connected component is relatively small but when it comes to index a video which contains about 1,200 connected components then the entire process might take several minutes to complete. At the retrieval time the only features computed are the features that describe the query, for everything else is much faster to load the 1,200 feature vectors from a file than computing them on the fly. Currently, the computed features are stored in hash tables indexed by the id of the video and then by the id of the sketch where these were found. However, no other special structure is being applied, but some more sophisticated structures could make it possible

to find the nearest neighbors of a symbol in faster times, and that is definitively part of the future work. After the connected components are added to the index, the next task is to generate the virtual key frames of the video by grouping content regions based on their life spans on the board.

5.2.3 Sketch Grouping Algorithm

The sketches formed by the content change detection algorithm are a good way to divide all the content of the board into small units that make the indexation process much easier than handling all connected components individually. However, since the creation of these regions is sensitive not only to spatial locations of the connected component but also to edition times, sometimes some elements that should be part of a single region get split into multiple regions due to long pauses of the speaker during the writing process. One way to overcome this problem is by generating groups of sketches that were present in the board at a given time. If the times used to generate these reconstructions of the state of the whiteboard are consistent with certain events on the video like massive erasing of content, then we could say that these groups represent the key frames of the entire lecture. One advantage of these key frames is that each of them can be used to describe an entire segment of the video. Another advantage is that they allow large queries to be completely matched using parts of different sketches that were present on the board at the same time.

The system needs to work with the time stamps stored along with each sketch in order to reconstruct the state of the whiteboard at a given time. Note that while the system could try to just extract a frame from the video at that given time, there are no guarantees that the speaker will not be present on the video on that specific frame. Since the sketches are extracted from different frames where the system detected that the speaker was not blocking that specific sub region of the board, then we could say that multiple frames are usually required to reconstruct the image of just the content of the whiteboard at a given time. Two time stamps are important for reconstruction of content, the first one is the last modification time as before that time the sketch was not present on the whiteboard or it was being edited, and the second is erasing time which tells the system when the sketch is no longer present on the whiteboard.

The process of reconstruction is done as follows: First, the system generates a sorted list that contains all edition and erasing times of all the sketches on the board. Each of these times represents a candidate time for a key frame insertion. Then, for each time stamp the system evaluates which sketches have reached their last modification time and adds the completed sketches to the list of current sketches. Then, the system also checks which sketches have completed their erasing time, if there are no new erased sketched then the system continues with the next time stamp on the list. When the erasing time of any of the sketches on the list of current sketches is reached by the algorithm, then the system will insert a key frame at the current time stamps which will contain a copy of the list of current sketches. The next step is to remove the erased sketch from the list of current sketches. Usually the speaker deletes entire sections of the whiteboard which results on many sketches with erasing times close to another. The system cannot insert a new key frame for every erasing event. Instead, the system will insert a new key frame on an erasing event if and only if new sketches have been added after the

last key frame insertion. At the end of the process, there might be sketches that were written on the board by the end of the lecture and never got erased on the video. If this is the case and new sketches have been added after the last key frame insertion, then the system will take this final group of sketches and will create the last key frame using the ending time of the video as the corresponding time stamp.

The algorithm of generation of key frames by grouping sketches is very simple and produces acceptable results on most of the cases. However, it is sensible to errors in time stamps of the sketches which become evident when certain sketches stop appearing on the key frames before they were erased on the video. This is due to some errors that occur when only small portions of the sketch are erased, and then the system assumes that the entire sketch was erased or that what remains does not represent the original sketch anymore, and for that reason registers an erasing time stamp earlier than the real one for those sketches. Even though such kind of errors occur, the results are still useful as ways to represent the content on segments of video and also to be used as retrievable units. Figure 31 contains some examples of key frames generated by the algorithm. It is also important to notice that since the sketches are extracted from different frames of the video some important variations in the lighting of the scene occur making the average color of the whiteboard darker or lighter on some specific regions of content, and that becomes more obvious on these key frames.

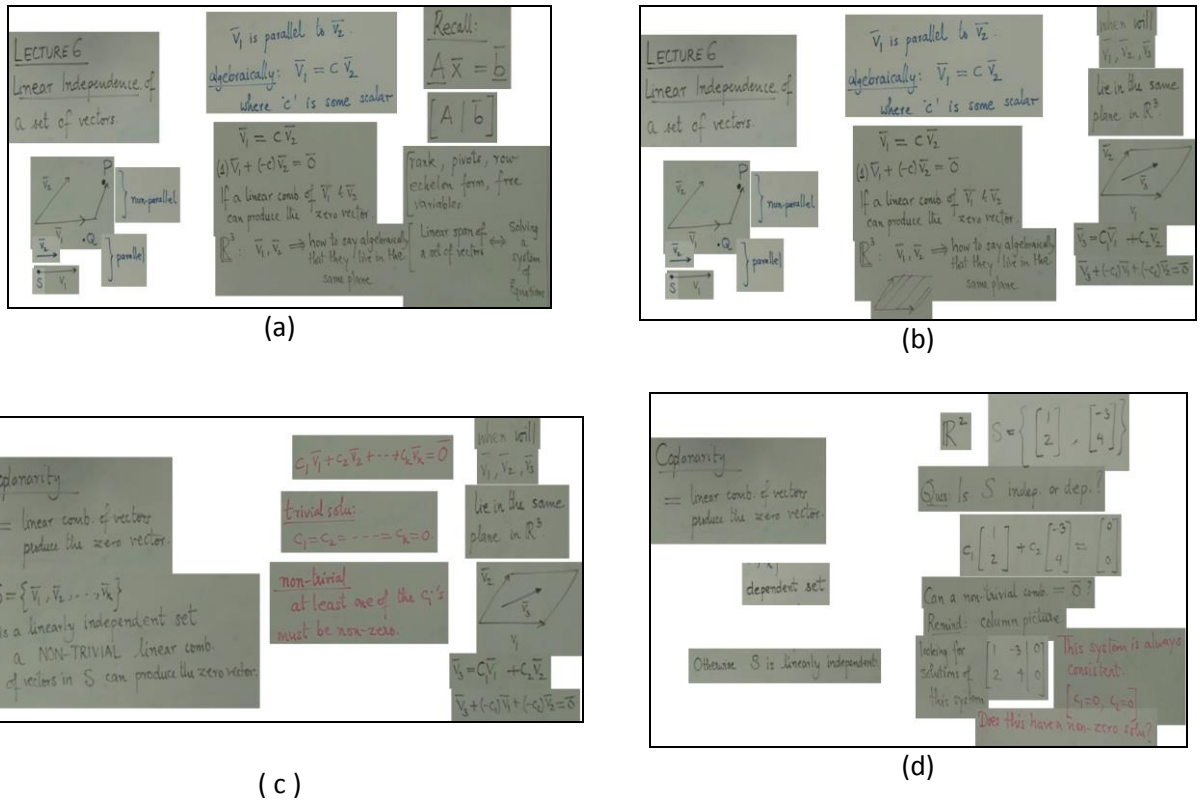


Figure 31. Examples of different key frames extracted from a lecture. Note how each of them shares at least one region with the previous key frames as usually the speaker erases only a portion of the board and keeps intact other parts of the content. In the case from (c) to (d) the speaker did erase the right side and wrote new content over it, but also edited part of the old content making the system believe that the entire content region was erased.

After the creation of the key frames, the system stores these groups of sketches as part of the index to avoid computing them again for every new query. These structures usually require small amounts of memory as they only represent groups of what is already stored on the index. At this point the system has indexed sketches with their corresponding connected components, and also the key frames. The next step is to describe the spatial structure of the sketches and key frames and add those descriptions to the index. This description of spatial structure is achieved through neighbor graphs as specified in the next section.

5.2.4 Description of sketch structure

The description of the individual connected components is very important on the current application because a good description can help the system to achieve good matches on the retrieval process. However, the best matches usually require more spatial information to be taken into account since math formulas are involved and the structure of an expression is a fundamental matter in math. There are many possible approaches on how to describe the structure of a sketch. A possible approach is using a hierarchy between elements like the method described in [36]. Another possibility is to describe certain relationships between neighbors and one or even many graphs to describe these relationships [46]. Through observation of the input data it was determined that isomorphism of sketches is not necessarily what makes pairs of regions to be related as it is shown in figure 32.

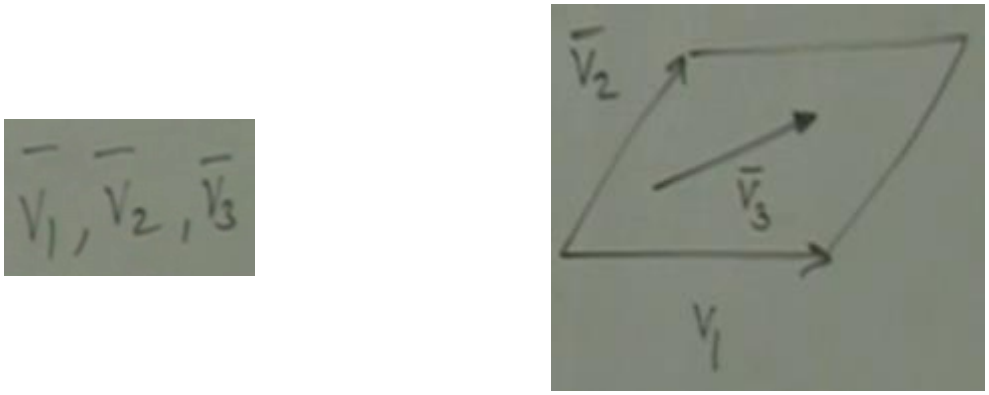


Figure 32. Example of two sketches that can be considered related to each other but have a completely different visual structure.

It is true that to some extend two regions need to share different elements to be considered matching regions. It is also true that good matches usually have a very similar structure, but it is also true that these structures do not need to be isomorphic to be potentially related. In the example shown in figure 32 it can be seen that two non-isomorphic structures can be considered related, and this is mainly because they have some isomorphic sub structures in common like for example the vector notation for v_1, v_2 and v_3. Based on this reasoning, one of the retrieval methods implemented works with neighbor graphs to define pairs of related symbols that should be matched between the query and the

content stored in the index. The generation of these graphs is better done at the preprocessing stage and the results are saved within the index file.

Graphs are defined on this application by creating a vertex per each individual connected component on the region being described. Then, the edges are created and their weights are assigned in terms of visual proximity between their corresponding connected components. Probably one of the most obvious methods to define visual proximity between connected components is using the Euclidean distance between the centers of the connected components. However, there are cases where using the distance between centers can make two connected components that are very close to look really far away. Another possible measurement of visual proximity is the distance between the bounding boxes of the connected components. This distance is defined by the smaller distance between sides of the bounding boxes of two connected components, and when these bounding boxes overlap the distance is equal to 0. The system currently uses a combination of both measurements of visual proximity.

The distance between all pairs of vertices is calculated and then the system filters edges between pairs of components that are not visually close. A way to know which edges to keep and which to remove is by applying a Minimum Spanning Tree (MST) algorithm over the graph. A MST is guaranteed to be the sub-graph that keeps all the vertices connected while minimizing the sum of the distance of all remaining edges. Depending on the measurement of distance between components used to create the graph, it is possible to obtain different MSTs for the same sketch. Since the goal is just to keep all edges between a vertex and its closest neighbors, the system can simply merge the edges found on each MST to form a neighbor graph. This graph will have a reduced number of connections while keeping a vertex connected to its closest neighbors as determined by the two measurements of distance. Figure 33 shows the neighbor graph obtained for the sketch of a matrix.

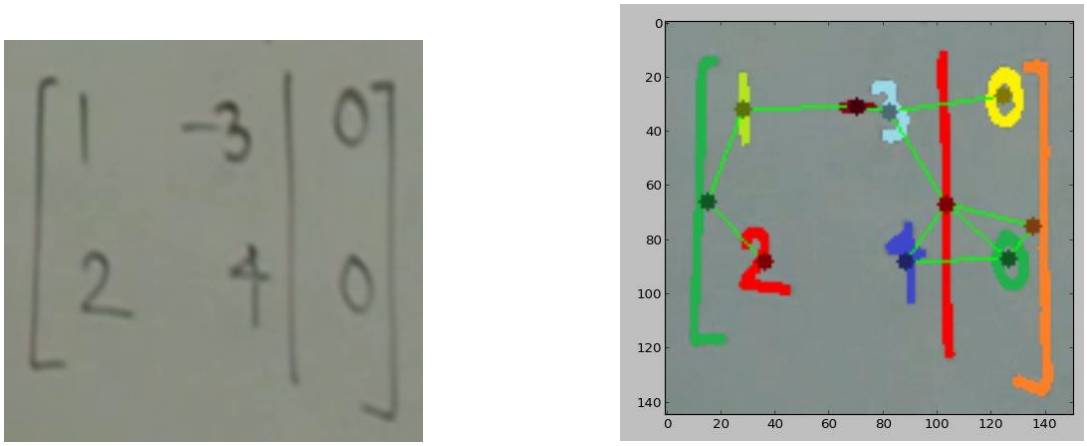


Figure 33. The graph generate for the sketch of a matrix. At the left the original sketch, at the right the graph obtained connecting the components that are visually close by distance between centers or by distance between the borders of the bounding boxes.

The same process applied for individual sketches is also applied to key frames to genera bigger graphs that not only connect the closest connected components inside of the sketches but also connect

the closest components between pairs of sketches found on the same key frame. This will allow a single query to match pairs of elements from different sketches located on the same key frame. Figure 34 shows the results of the graph generation algorithm over a key frame.

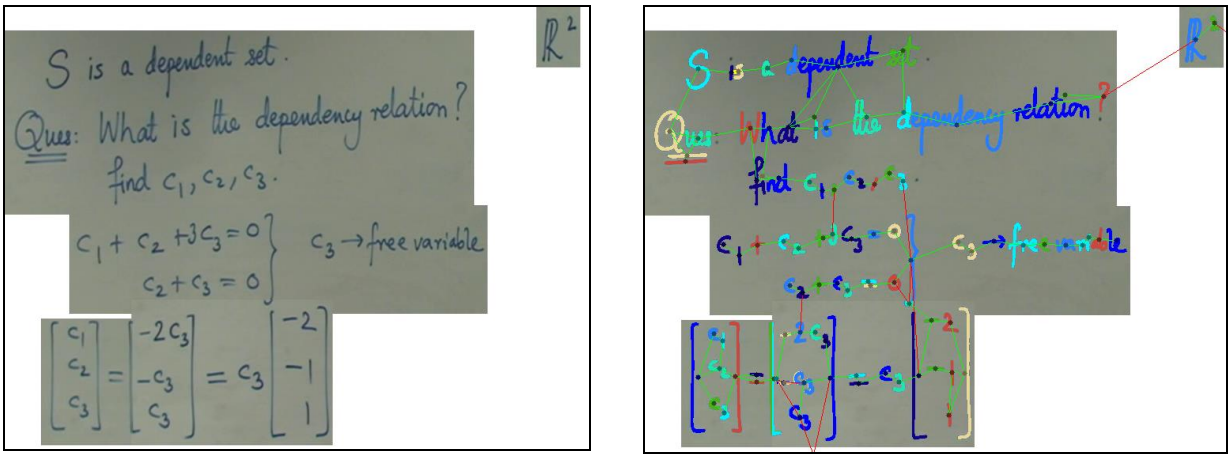


Figure 34. Neighbor graphs formed on a key frame. The left image is the original key frame, the right image is the graph formed for that key frame. The green edges represent standard edges between connected components of the same sketch. The red edges represent edges between connected components from different sketches.

The graphs are the last kind of the structure computed and stored on the index. Once that all graphs have been computed for the sketches and key frames of a video, these are stored for later retrieval processes. At this point the video is completely indexed and the content found on it can be retrieved using the different sketch retrieval methods implemented which are described on the following section.

5.3 Sketch Retrieval

After the content of all videos in the collection has been indexed, the system is ready to accept queries. A query in this system comes in the form of an input image or sketch, and the system will try to find either sketches or key frames that seem to have similar content. The current system is recognition free and all matching is done in terms of visual similarity between the query and the stored content. For testing purposes, the system allows the user to specify one sketch from the index as the input query. If the user does not specify a query, the system can also select a random query from the stored sketches. On the current scope the system is not allowing external images to be used as queries, but this is a feature that could be included as future work.

All content on the index is stored in forms of individual sketches and also in the form of groups of sketches or key frames. The different retrieval methods were developed to allow the search by the

two kinds of content. However, there are some methods that work with sketches but lack the scalability required to be applied at the level of key frames. When the search is limited to the level of individual sketches, it has the advantage that some of these non-scalable methods like the search by F-Measure works fine because of the small numbers of connected components. The disadvantage of looking for individual sketches is that divisions of content are not optimal and the query could not be matched if the same content is found in parts of different sketches. When key frames are used, the greatest advantage is that the query can be completely matched by partially matching different sketches on the key frame. The disadvantage is the increased running time because the number of possible matches is higher, and this is due to the fact that some sketches belong to multiple key frames increasing the total number of comparisons required for this type of search.

More than just finding specific images, the general idea is to use the time stamps of the images found to relate them to the specific segments of the video from which they were extracted. In general, sketches represent very short regions of the video generally measured in seconds while key frames represent larger segments of the video generally measured in minutes. It is out of the scope of this project to create the user interface required to visualize these segments of videos. However, the current system could be easily modified in the future to interact with an external application that displays the segments of video as final results. The next sub section explains the different measurements of similarity that were implemented for retrieval of sketches and key frames.

Measurements of similarity applied for retrieval

The method used to measure the similarity between two given regions of content is really important and it will define the performance in terms of quality of results and running time. Different methods were tested and some of them produced interesting results while other did not, and these methods were: Count of hits by nearest neighbors, recall of connected components matching, F-Measure of connected components matching, recall of SURF matching, and recall of pairs of neighbors matching. All these method produce a score of similarity between regions, and then the resulting values are used to rank the regions stored in the index finally retrieving only the best N matches, where N is a number defined by the user. Each of these methods is explained here.

Count of hits by K nearest neighbors

This was the most naïve approach implemented and it is only applied on sketch versus sketch matching. The basic idea is simple, take all of the connected components on the query, and for each of them get the first K nearest neighbors whose distance is below a given threshold. For all of those K connected components, a hit point is assigned to the sketch to which they belong. If the query has 10 connected components, and K is set to 100, then 1000 nearest neighbors will be obtained in total which will generate 1000 hit points distributed among the different sketches where those components were found, and then the sketches are ranked by their number of hits in descendent order. The method is very simple and since it does not count for unique matches, it usually gives really high scores to some regions if they contain several nearest neighbors of a single connected component of the query. Some of the results obtained can be considered good matches, but it will also retrieve many unrelated things just

because they contained several copies of one or more elements found in the query. In the other hand, the method is really fast and it was a good point to start the process of retrieval of sketches. Given that repeated connected components is a problem for this method, testing it over key frames would just have made the problem worst because they are likely to contain even more copies of each component.

Recall of matched connected components

The second method was designed to correct one of the biggest problems of the previous one, and it is that a single connected component of the query was being matched multiple times by many similar connected components on some large sketches. This resulted on some regions with higher scores than the actual level of visual similarity perceived. To correct this problem, the next goal was to ensure the uniqueness of the matches which means that one connected component from the query can match at much exactly one connected component of another region. If a connected component on the query can only match one connected component of the tested region, then the recall of matched connected components can be obtained dividing the total number of matches by the number of connected components on the query.

$$Recall = \frac{\#Connected\ Components\ Matched}{\#\ Connected\ Components\ on\ Query}$$

Note that this method only ensures that a connected component on the query will be matched by exactly one connected component on the tested region, but it does not ensure that a connected component on the candidate region will not match multiple connected components of the input query. In other words, the previous method allowed relations many-to-many between connected components on the query region and the candidate region, but this second method just allows one-to-many. The search method was test with sketch versus sketch matching and sketch versus key frames matching, but since a better method was found for sketch versus sketch matching, it was kept only for sketch versus key frame matching. It is also fast and usually produces better results than the previous method.

F-Measure and the assignment problem

A more accurate version of the previous method is obtained by replacing the recall by the F-measure or F-Score of the connected components. The F-measure is the harmonic weighted average of the recall and the precision. Recall is measured in terms of components matched against the total of connected components in the query sketch while precision is measured in terms of components matched against the total of connected components in the candidate sketch. However, the only way to get the exact the recall and precision is to modify the matching system to force it to accept one-to-one matches only. Ideally these one-to-one matches should be made in a manner that the sum of the total distance between matched connected components is minimal. To obtain that minimal distance, the entire matrix of distances between all candidate matches can be calculated, and then the Hungarian method can be applied to solve the assignment problem [38] over that matrix. This method requires the distances matrix to be squared, and since the number of connected component on the query and on the candidate region are going be different in most of the cases, some padding columns or rows must be added using a distance value that is higher than any other expected distance between connected

components. Note that all the assignments that include values found on those padded columns or rows imply that no real matching was found for that connected component. When all the assignments have been computed, the system can apply a threshold of distance and accept only the matches with a distance below the threshold. This final number of assignments is used as the numerator of both precision and recall, and then the f-measure is finally computed. A value of 0.0 is the worst value for an F-Measure while 1.0 is the best value or a perfect match.

$$Recall = \frac{\#Connected\ Components\ Matched}{\#\ Connected\ Components\ on\ Query}$$

$$Precision = \frac{\#Connected\ Components\ Matched}{\#\ Connected\ Components\ on\ Tested\ Region}$$

$$F - Measure = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

While the results obtained by this method seems very interesting, it is way too heavy to be applied in practice as a single query can take from a few minutes to even a few hours. The problem is that the assignment problems has a complexity of O(N^3) on its best implementation, and since some sketches have as much as a 100 connected components the matching between those sketches takes many seconds which is too much for just comparing a pair of sketches. What is more, the approach cannot be scaled to be used on sketch versus key frames matching because the total number of connected components that is usually found on a single key frame is much higher than just 100 connected components.

SURF

The SURF were successfully applied to visually align the content between the two sources of video on this project, and one experiment was to use them as a way to find queries inside the key frames. The SURF work by finding key points on two images that will be compared, and then these key points are matched to find one image inside of another. However, not all pairs of matches are really good matches and the system need several good matches to be found before it can really find an image inside of another. Ideally, a high percentage of good matches should mean that the image has been found, but in practice many key points can be matched on the current data and that does not mean that the candidate region contains the query. As a test to define a method for retrieval using SURF, the recall of key points matched is being used to rank the similarity between pairs of sketches and between a sketch and a key frame. The method works for perfect matches, which is when the query is present on the candidate region, but it usually does not work for partial matches making this method a bad option for retrieval as most of the related content for a query will be partial matches.

Something good about these features is the fact that they can be computed and matched really fast. For example the queries presented in the results section took less than 10 seconds to be executed which is really fast considering that these features were not stored on the index and the system had to calculate them on the fly as the query was being executed. If the key points and their SURF were store

on the index, the execution time would be really small, but the reason for not adding these to the index was the unacceptable results obtained on different queries tested.

Recall of matched pairs on neighbor graphs

The last method developed for retrieval uses the neighbor graphs obtained during the indexation process to attempt to match partial structures. One of the common problems of the previous methods is that when they match connected components from the query on the candidate region, they usually match any connected component regardless the actual relationships between them, and that produces undesirable matches like for example the number 32 matching 2^3 because the individual connected components are the same. A partial solution to this unrestrained matching problem is to add some basic restrictions by matching pairs of connected components with the same spatial relationship. Applying this restriction in the previous example would mean that a 3 and a 2 would only match if another 3 is found along with another 2 with the same spatial relationship between them. In a sketch with 10 connected components there will be 45 potential pairs of elements to consider, but since the neighbor graphs are being calculated first, then probably the number of pairs to use for the actual matching will be reduced to about only 12 pairs making this method scalable even for large graphs.

The spatial relationship between two connected components that are linked in the neighbor graphs is defined by the angle formed by the location of the center of one of them relative to the center of the other one. This angle basically defines an orientation between the pair regardless of which of the connected components is being used as the center of reference. The idea is that matching will be limited only to pairs that have almost the same angular orientation with some small degree of tolerance for variation. If a pair of connected components on the query has certain angle orientation, and a pair of connected components on the candidate region has an angular orientation close enough to the orientation of the first pair, the system will compare the corresponding connected components to get a measurement of total distance by averaging the distances obtained between the pairs of corresponding connected components. Figure 35 shows an illustration of this process of matching edges.

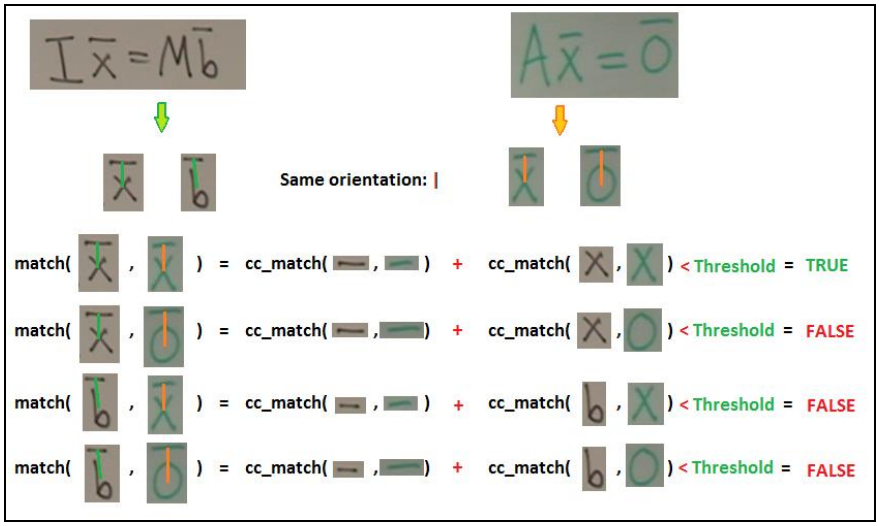


Figure 35. The process of matching edges to estimate similarity between sketches.

The total measurement of distance between a query and a candidate region, either a sketch or a key frame, is obtained using the recall of pairs of edges on the graph of the query that were matched with edges found in the candidate region. Note that there are no further spatial restrictions applied between these edges and as a result it can be the case that two edges that have a vertex in common on the query could match two edges with no vertex in common on the candidate region. However, in practice the results obtained with this method are still much better than the ones obtained using completely unrestrained matching of connected components.

The running time required for this matching is higher than the time required by some other methods as each single connected component is tested for matching more than once. Nevertheless, this running time is not that bad and can be improved by using a more sophisticated method of indexation optimized for matching edges.

6. Results

The different methods were tested against predefined queries to compare the results that each of them would return for the each of them. As expected, these results were usually very different between methods as each of them ranks similarity using different criterions. All of the tests described on this section were executed on a laptop running with windows 7, with 8 GB of ram and Quad core processor of 2.4 GHz with turbo boost up to 3.1 GHz. Even though most of the methods allow parallelization, this was not implemented and all running times are measured on all comparisons being made on a single thread. The index file used for all tests contains the information of 6 lectures for a total of 96 key frames, 543 sketches and 8,732 connected components.

Using count of hits by K nearest neighbors

One example of a query using the method of counting hits by k nearest neighbors with its top 5 results can be found in figure 36. The total of nearest neighbors K was set to 1000. That amount of nearest neighbors was selected to ensure that the search would limit the final number of nearest neighbors retrieved based on the threshold of similarity instead of just the top K nearest neighbors. The running time was only 3.52 seconds for this query with 10 connected components.

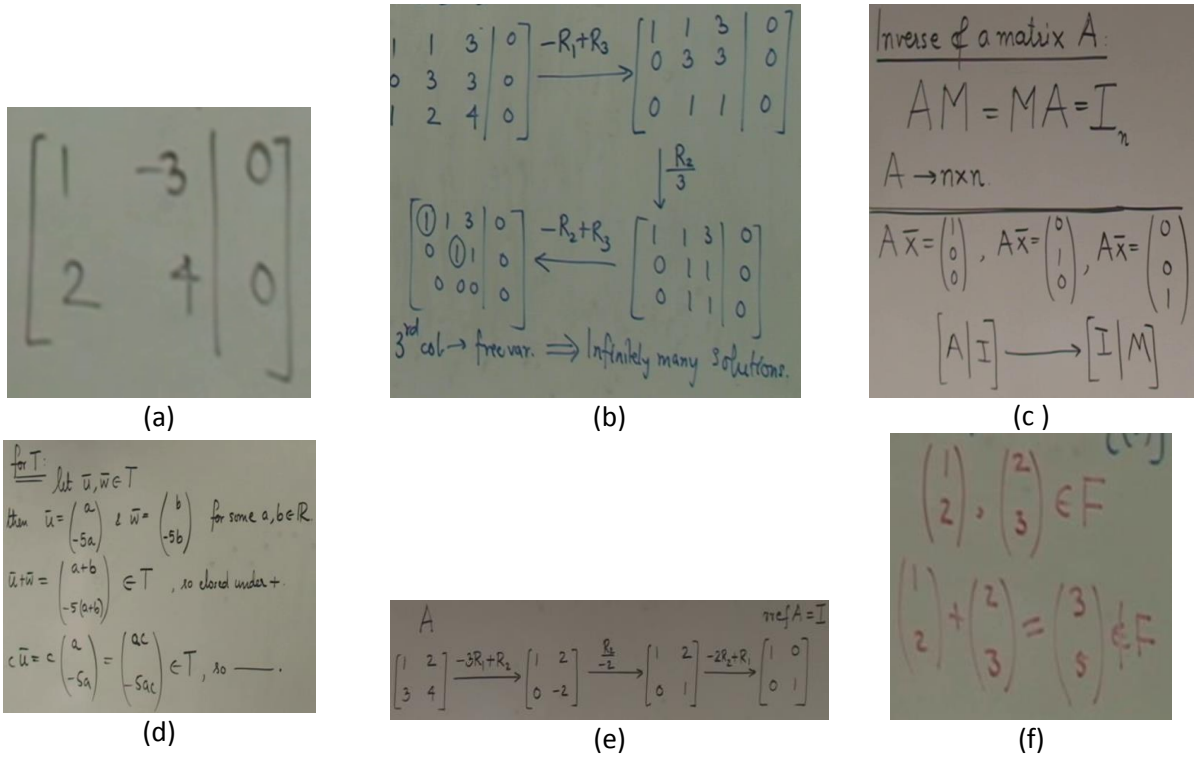
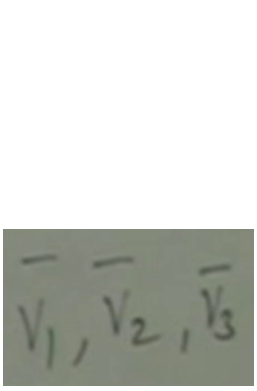
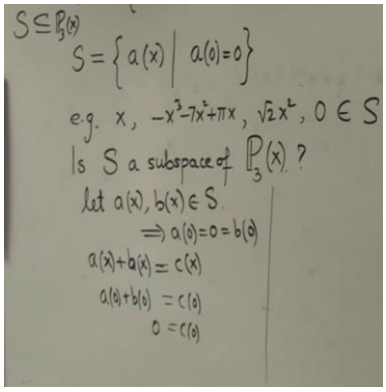


Figure 36. Query executed using the counts of hits by K nearest neighbors: (a) The input query, (b) First match: 78 hits, (c) Second match: 49 hits, (d) Third match: 43 hits, (e) Fourth match: 38 hits, (f) Fifth match: 33 hits

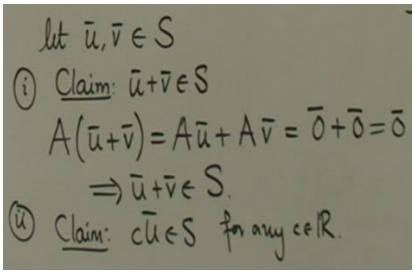
A second query is shown in figure 37. This one took 3.27 seconds in running time. As it can be seen, some of the retrieve results can be considered related like figure 37.d, but others do not seem related at all like for example figure 37.e. Note this last one was probably high ranked by the large number of horizontal bars that it contains matching the horizontal bars on the query.



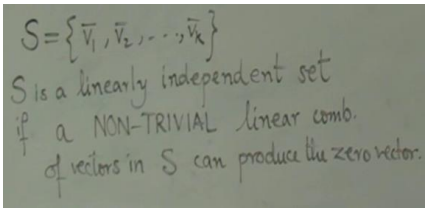
(a)



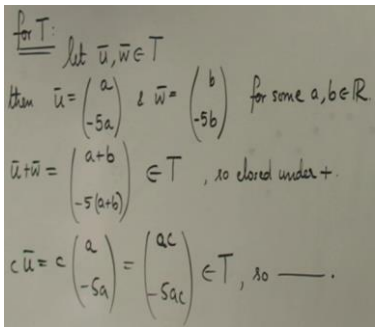
(b)



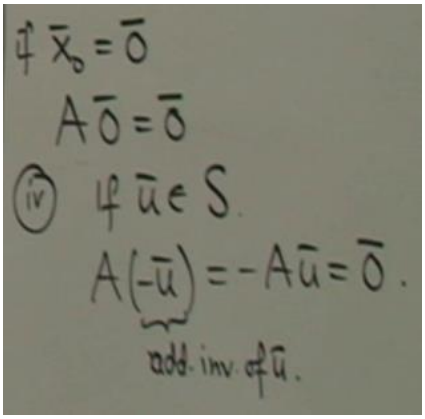
(c)



(d)



(e)

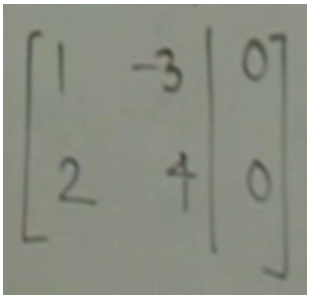


(f)

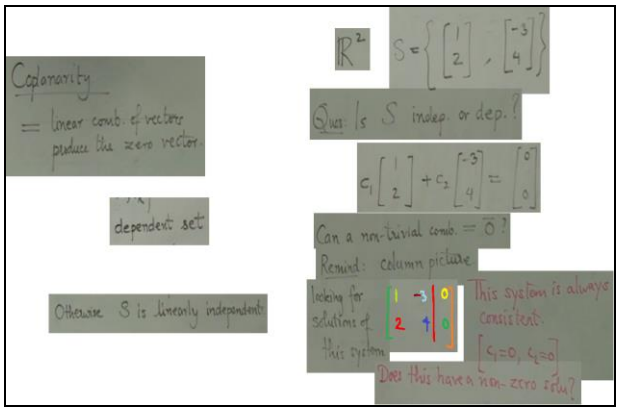
Figure 37. Query executed using the counts of hits by K nearest neighbors: (a) The input query, (b) First match: 29 hits, (c) Second match: 28 hits, (d) Third match: 27 hits, (e) Fourth match: 26 hits, (f) Fifth match: 24 hits.

Using recall of matched connected components

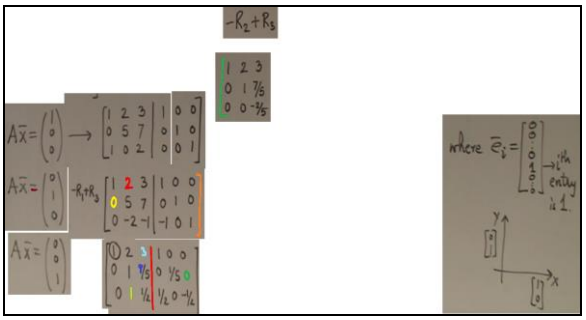
Figure 38 shows an example of the application of this method for the same query of figure 36 but this time the search is done with sketch versus key frame matching. The time required to execute this query was 5.88 seconds. Since a single sketch can be included in multiple key frames, 2 out of 3 of the top 7 best matches are excluded here because all of them contained the query sketch. Additional coloring is added to visualize the connected components being matched. Note that since there are no structural restrictions for the matches, the system will retrieve key frames as long as they contain many of the connected components found in the query independently of the positions at which they are located resulting on many matches that at first glance seem to be unrelated.



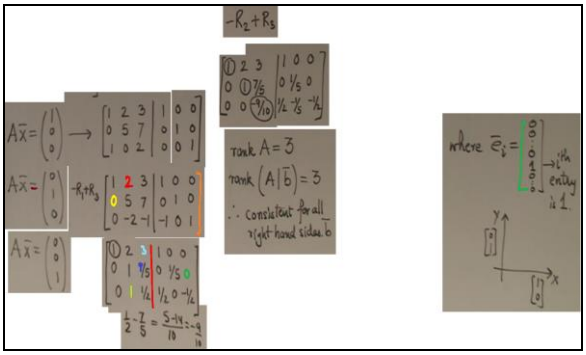
(a)



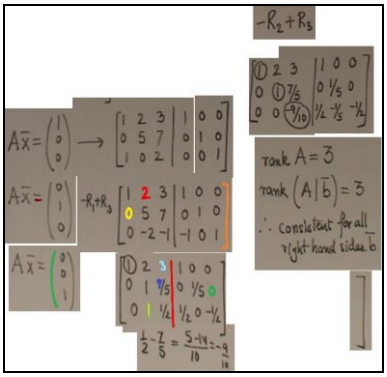
(b)



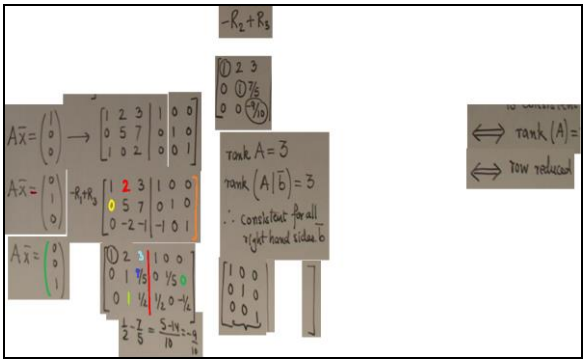
(c)



(d)



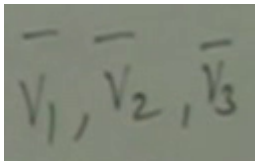
(e)



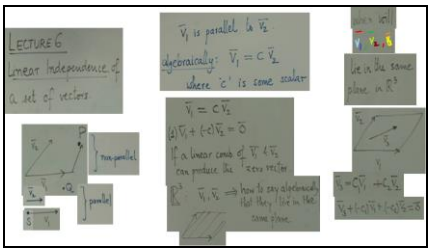
(f)

Figure 38. Results for Sketch versus Key Frame matching using the Recall of matched connected components: (a) Input Query, (b) Top 1 match: 100% recall (contains the query), (c) Top 4 match: 100% recall, (d) Top 5 match: 100% recall, (e) Top 6 match: 100% recall, (f) Top 7 match: 100% recall.

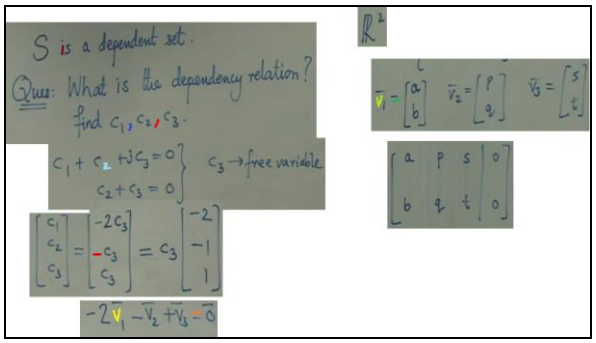
A second query is shown in figure 39. This second query took 5.84 seconds. Again, 4 out of the top 5 matches are not present here as all of them contained the input query. Similarly to the first query shown, most of the results seem to be unrelated because connected components are being matched without spatial restrictions.



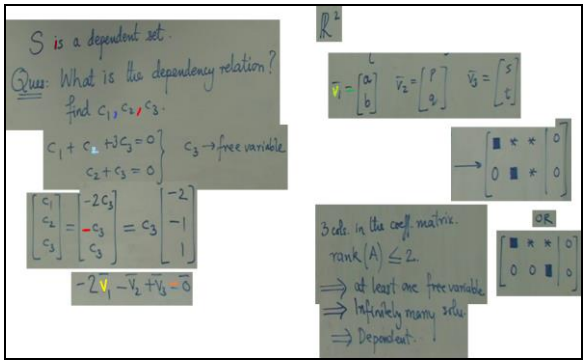
(a)



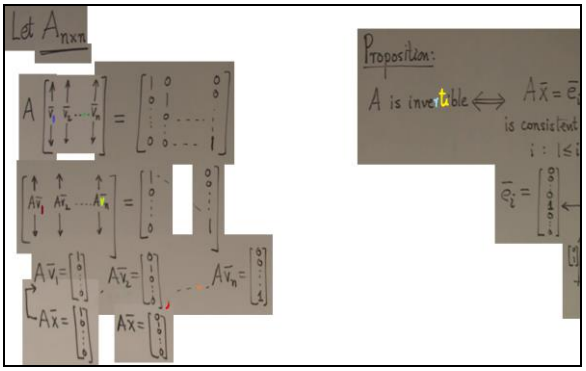
(b)



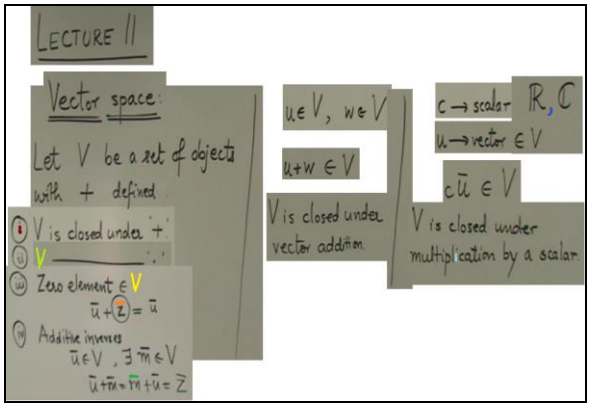
(c)



(d)



(e)



(f)

Figure 39. Results for Sketch versus Key Frame matching using the Recall of matched connected components: (a) Input Query, (b) Top 1 match: 100% recall (contains the query), (c) Top 6 match: 90% recall, (d) Top 7 match: 90% recall, (e) Top 8 match: 90% recall, (f) Top 9 match: 90% recall.

Using F-Measure and the assignment problem

Figure 40 contains an example of a query using the F-Measure. It took 3 minutes and 7 seconds to complete this query which is 37 times the time that Recall method required for matching the same sketch versus complete key frames. The average time required per single comparison is usually small, but the problem is the complexity of the assignment algorithm which makes it take several seconds on certain sketches with a high number of connected components.

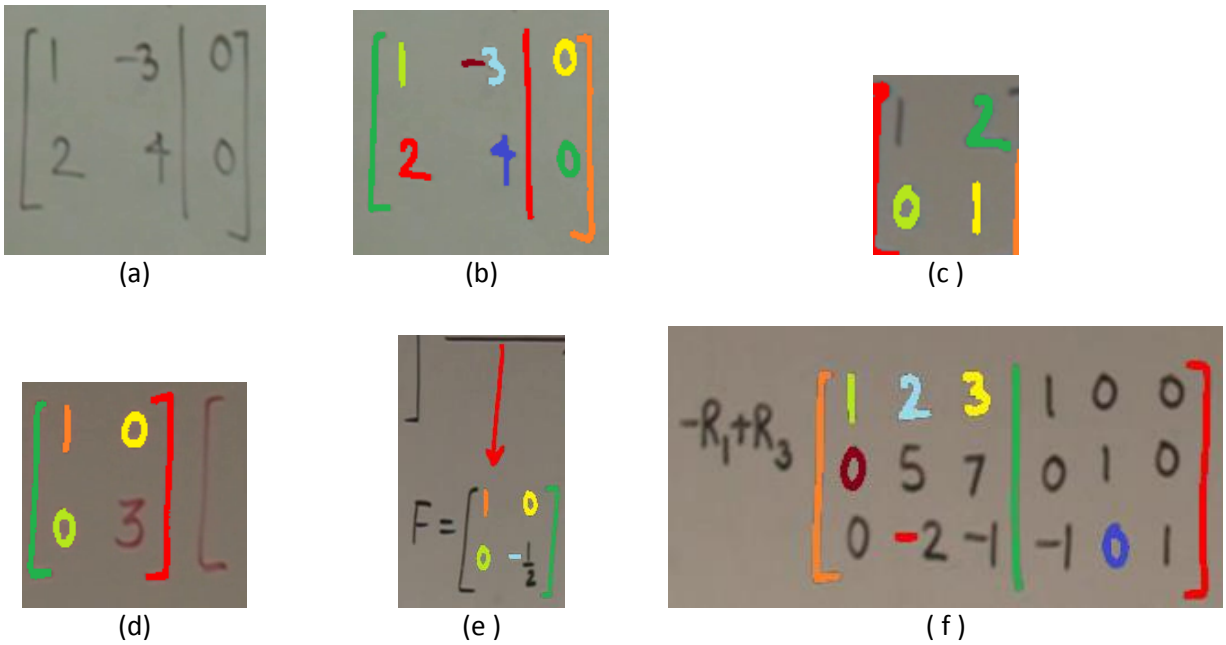


Figure 40. Results of a query using the F-Measure of matched connected components: (a) The input query, (b) Match #1, F-Measure: 1.0, (c) Match #2, F-Measure: 0.58, (d) Match #3, F-Measure: 0.58, (e) Match #4, F-Measure: 0.52, (f) Match #5, F-Measure: 0.51.

The results obtained by the F-Measure on the query shown in figure 40 reflect that forcing the system matches to be one-to-one is good. However, since no structural restrictions are being applied to these matches, some regions can be matched with a high F-score as long as they share many connected components independently of their structure. Another query is show in figure 41 which took 3.11 minutes to complete. The fact that returned regions are also smaller than using the previous measurements is thanks to the fact that F-measure considers precision and recall at the same time.

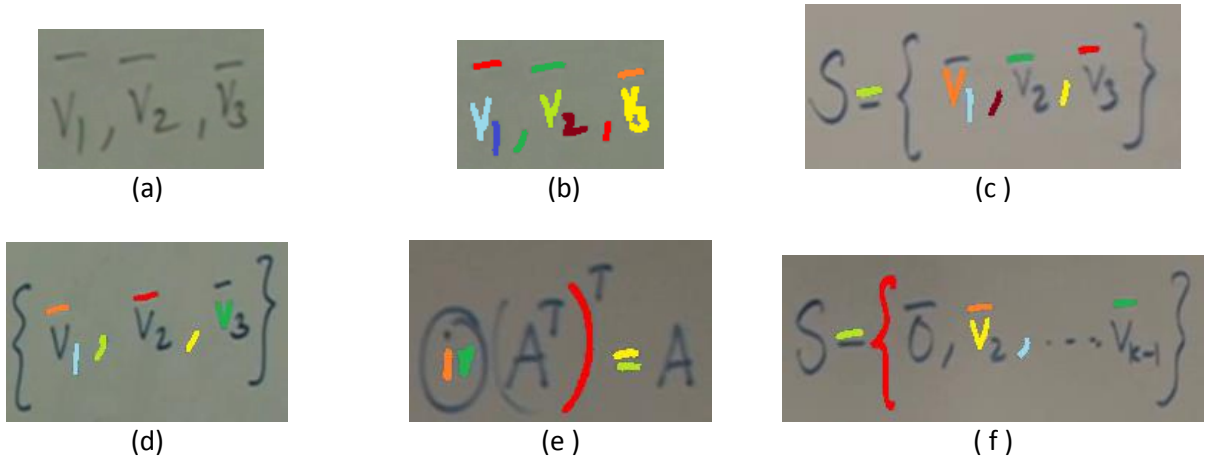
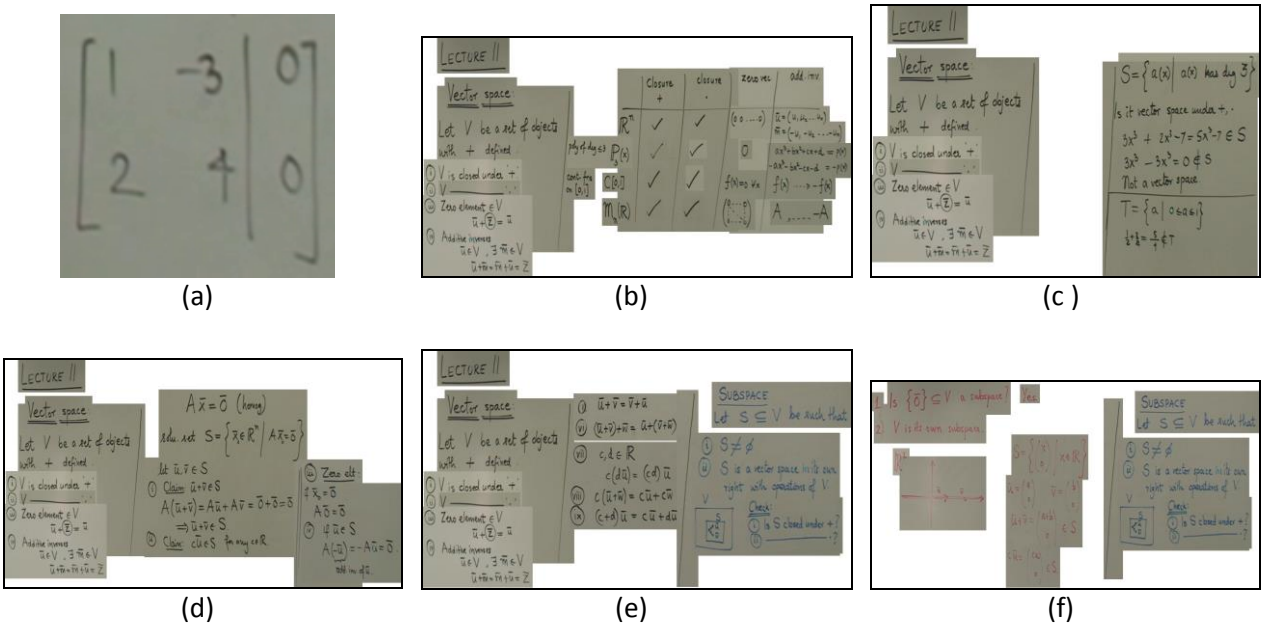


Figure 41. Results of a query using the F-Measure of matched connected components: (a) The input query, (b) Match #1, F-Measure: 1.0, (c) Match #2, F-Measure: 0.56, (d) Match #3, F-Measure: 0.52, (e) Match #4, F-Measure: 0.47, (f) Match #5, F-Measure: 0.44.

Using SURF

Figure 42 shows some results for a query using the recall of the SURF matches. It took 9.95 seconds to run this query and the results are not acceptable.

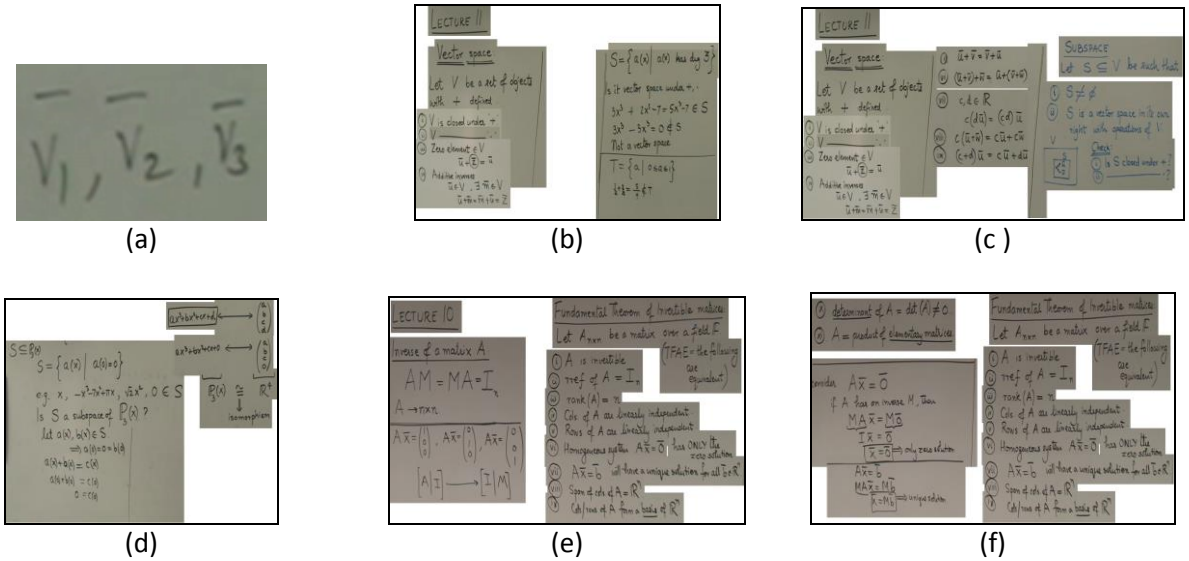


(a) (b) (c)

(d) (e) (f)

Figure 42. Example of a query using SURF: (a) Input Query, (b) Match #1, Recall: 100%, (c) Match #1, Recall: 100%, (d) Match #1, Recall: 100%, (e) Match #1, Recall: 100%, (f) Match #1, Recall: 100%.

A second query is shown in figure 43. This one took 13.10 seconds and results are also bad.



(a) (b) (c)

(d) (e) (f)

Figure 43. Example of a query using SURF: (a) Input Query, (b) Match #1, Recall: 100%, (c) Match #1, Recall: 100%, (d) Match #1, Recall: 100%, (e) Match #1, Recall: 100%, (f) Match #1, Recall: 100%.

Using Recall of matched pairs on neighbors graphs

Some results using this method can be seen on figure 44. The running time required for this matching was 17.36 seconds.

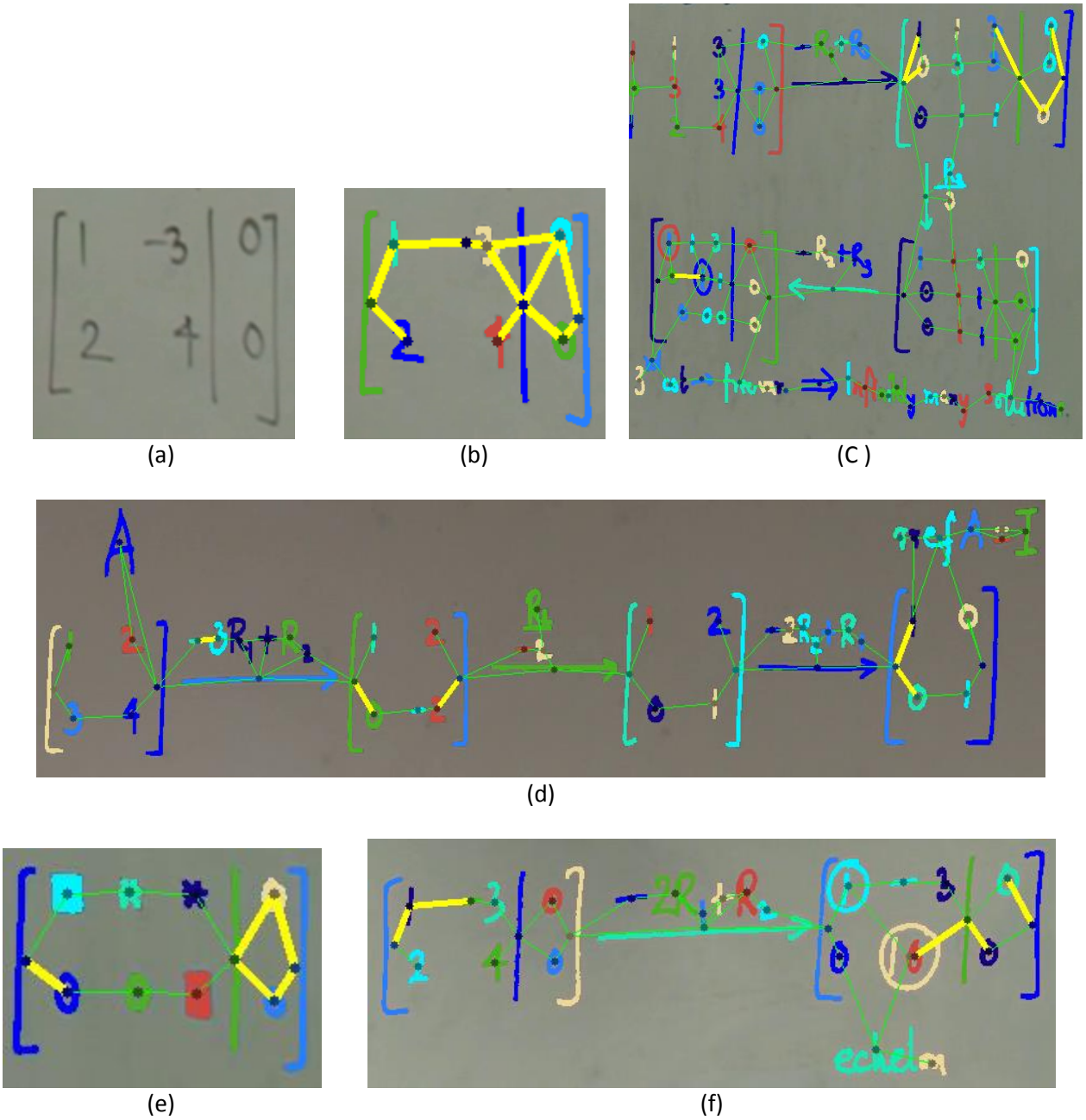
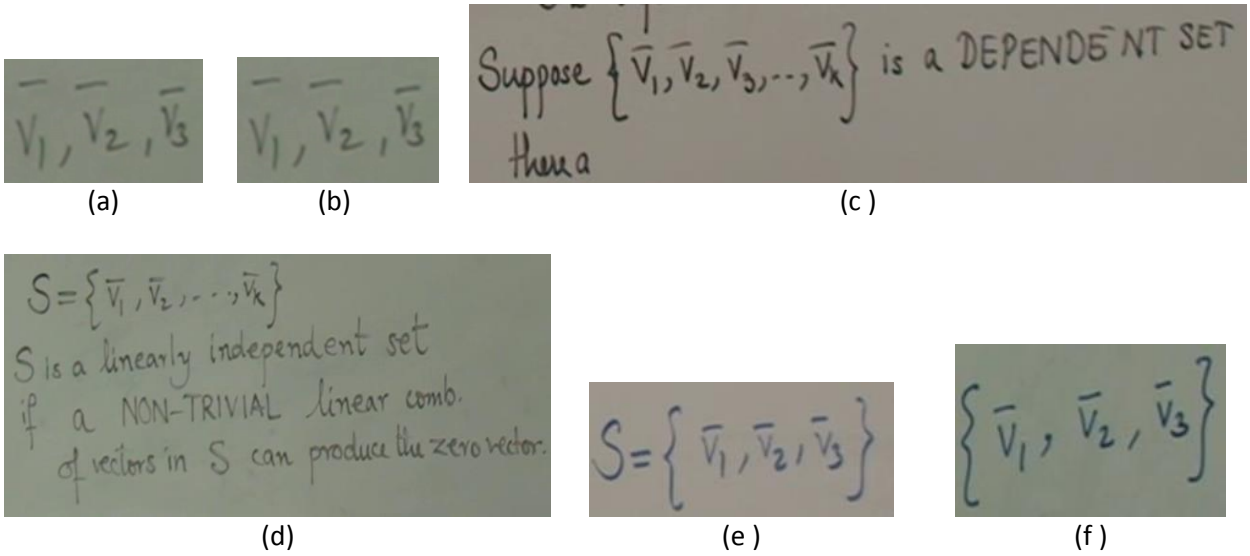


Figure 44. Query executed using the recall of matched pairs on neighbor graphs: (a) Input query, (b) Match #1, Recall = 100%, (c) Match #2, Recall = 63%, (d) Match #3, Recall = 45%, (e) Match #4, Recall = 45%, (f) Match #5, Recall = 45%.

It is worth to mention that the input query used for the different tests to compare all methods is actually hard to match in terms of meaning as it simply represents a matrix. The results of the recall of matched pairs on neighbor graphs can be considered the best because the images retrieved by the system were matrices as well, but more important matrices that contains combinations of the same elements most of the time with the same arrangement as for example some matrices that also contain zeros on the right side.

Another example of this method applied is shown in figure 45. This time the query simply contains the notation for three vectors. Note that all the top matches contain vectors, and even if they have different arrangements, they can be considered as valid matches because in most of the cases they even have the same sub-indices for the vectors.



(a) (b) (c) (d) (e) (f)

Figure 45. Query executed using the recall of matched pairs on neighbor graph: (a) Input Query, (b) Match #1, Recall: 100%, (c) Match #2, Recall: 66%, (d) Match #3, Recall: 66%, (e) Match #4, Recall: 55%, (f) Match #5, Recall: 55%.

Currently there are many errors in the matching process specially pairs of edges that should have not been matched and others that could have been matched and they were not. Also, the method is very sensible to cases where touching symbols become a single connected component and therefore something that should have been matched as a pair now can only match be matched as a single connected component. The confusion errors can be removed or mitigated with further refinement of the parameters for the matching of pairs. Also, additional features that describe the pair as a single connected component could help to refine that matching or even allow matching with touching symbols that became a single connected component. However, even with those current limitations, the system achieves acceptable results for many queries making it the most promising method of all the methods tested so far, and also a good start for future work.

7. Discussion

Five different methods to measure the similarity between queries and candidate regions were implemented and tested. Some of them produce meaningful results while the others not always produce meaningful results. The most naïve method, the count of hits of nearest neighbors show reasonable results considering the simplicity of the method, however there were more meaningful sketches on the index that other methods retrieved on their top 5 matches. The fact that the highly rated regions on the count of hits of nearest neighbors are usually regions that contain many copies of elements that are part of the query reflect that allowing one element from the query to match multiple elements on the candidate regions is not a good idea.

The second method, the recall of matched connected components, showed more reasonable results than the previous method. The fact that one element in the query counts for only one hit per candidate region greatly affected the weighting system. In the other hand, the results shown that measuring similarity only in terms of the components from the query that were matched resulted in many regions with very similar scores. For example, the top 5 results for the first query show on figure 38 got a 100% of recall which means that since all of them contained a matching element per each element on the query they got equally ranked, and that also means that the final order on which they were listed was actually defined by the order on which they were matched, but all of them got the same score and none was considered a better match than the others. From the perspective of the user not all of them are good matches, if any of them can be considered a good match. The total number of additional elements present on the sketches retrieved is one of the first elements that a human would use to give a lower rating to some of these best matches.

The third method greatly improved the results by taking into account not only the recall but also the precision using the combination of both through the F-Measure. This, however, was done at the expense of using the Hungarian method to solve the assignment problem to make the matches of connected components unique. This algorithm has a high complexity and results on non-scalable running times if the number of connected components of a region is above 100 connected components. The effect of using the F-Measure is that when two sketches have the same recall, they will be ranked by highest precision which means that the system will prefer matches of about the same size of the query or in general containing only a few additional unmatched elements. Probably, using a less optimistic assignment of matches would result in much faster running times at the expense of losing some accuracy for a few queries that will count less real matches than the optimal number of matches that can be found using the Hungarian method. In the other hand, the best matches are still unrestricted in terms of structure of the query, and for that reason the results with this method will never be as good as the results obtained by methods that consider structure on the similarity measurement.

The method that applies SURF for matching of regions seems to be the most helpless of all methods tested. The problem is that the SURF were made to match key points and the best matches for these key points are not always real matches. In fact, the number of bad matches is usually very high making the recall of good matches a bad measurement for similarity. Also, these features were designed

to match objects on images even if they are rotated which means that it will produce many matches of key points that seem to be the rotated version of something from the query, but for the purpose of math retrieval they do not represent a good match. For example, horizontal lines can be matched with vertical lines using SURF, but in the context of math the orientation of a line gives it a specific meaning.

The last method tested, recall of matched pairs on graphs of neighbors, is the one that produced the most satisfactory results, and this is because it applies a more restricted level of matching by matching elements in pairs. The orientation of the line connecting the centers of the bounding boxes of two connected components in a sketch is being used to filter possible matches which means that even if a candidate region and a query contain the same connected components, but not a single pair is arrange the same way in the two regions, then it is possible that the total of pairs matched becomes 0 and such region would not be retrieved. Note that the example on figure 45 is a query containing notation of three vectors, and the top 5 results only contain regions that had vector notation on them, and these did not necessarily had the vector notation arranged in the same way, but because most of the pairs of elements of the vector notation were matched independently, then these regions were considered the best matches. In cases like vector notation, where specific pairs have a meaning by themselves, matching pairs as if they were single units will produce better matches than other methods that do not consider structure at all.

8. Conclusions

Extracting information from videos and then making this information available for retrieval can be a challenging task prone to errors at many steps of the process. Noise is a common factor on all steps and therefore special considerations have to be taken on every case. The procedures used for extraction of the information from its raw sources need to be very robust to work for most of the cases even for the noisiest ones. Also, variation is another important element that must be always considered in order to produce these robust methods. For example, on the current application two different methods for visual alignment were implemented, the first one worked fine for the first half of the dataset, but after the second half of the videos was introduced, the method stopped working correctly. A second more robust method for alignment was introduced which accounted for higher levels of variation. If such levels of variation had been considered from the very beginning, probably the first method would have never been used and the second method would had to come first. However, it is usually hard to visualize all possible kinds of noise and variations that come with the data from the very beginning and for that reason sometimes is better to create an initial naïve implementation that works for a few cases, and progressively increase its robustness as soon as variations and noise are identified.

After all information has been extracted, the method employed for indexation is really important because while some methods require more time in the indexation process, they can also speed up the retrieval process by allowing faster matching. The index is also dependent on the method selected for matching as different methods would require different elements to be pre computed and available at the time of retrieval. As long as new methods for similarity measurements are tested, the structure for the index cannot be considered to be final.

Many researchers have identified something that they have called the semantic gap [16] which in the current application means the difference between the best matches as defined by the system in terms of features and the best matches as defined by the user in terms of the meaning of the content matched. This is one gap that cannot be closed unless the system becomes smart enough to understand what the user is expecting to find with the current query. Of course, to make the system to understand human semantics behind a query is something that would not happen even after many years of improvements of the system.

In the other hand, different matching procedure can make the semantic gap larger or smaller depending on the level of the features of the query being considered by the system. The different features that can be extracted from a region of content can be considered as low level or high level based on what they describe. An example of a low level feature in this context would be the features used to match individual connected components, while something on a higher level could be the actual structure of a math expression. At the highest level we could say that a given query represents one specific topic in math, and as long as the system is not able to understand this kind of very high-level relationships, it is impossible to achieve the highest accuracy for an information retrieval system like the one presented on this report.

Future work

Different task have been identified as open for further improvement on the developed system. The most obvious one could be the retrieval task as the system can improve in terms of running time and quality of the results. More sophisticated methods for matching of sketches need to be tested, and just as it was found that restrained matching is better than unrestrained matching, the new methods have to consider additional restrictions that ensure more meaningful matches. In the last method tested for retrieval, the recall of matched pairs on neighbor graphs, it might be possible to achieve better results if chains of matched edges are forced instead of accepting the first matches of each edge as the best match. A candidate region with large chains of edges matched can be considered much related to the query. For example, two candidates can have the same measurement for recall of matched pairs, but if one of them has larger chains of matched pairs than the other one then it should be considered a better match. Also, while the current features are able to match similar shapes, these could be modified or replaced in order to get higher accuracy in matching and/or faster comparisons that reduce the execution time of the queries. In addition, it was noticed that with the exception of the method of counting hits, the measurement of distance between a query and a candidate region is independent of the measurement of distance for other candidate regions, and therefore these measurements could be executed in parallel to reduce the total time required to match a query against the entire database.

Other tasks that need to become more robust are the tasks that handle noise because independently of the method used for matching, the retrieval results will have limited improvement if the presence of noise is not reduced. For example, the additional connected components on extracted regions could not be removed using the proposed method because of the noise on the Mimio video, but a different procedure could be able to remove it even if the Mimio video is noisy. For that purpose, an interesting change could be to implement the detection of content changes using the main video instead of the auxiliary video. Also, the problem of the touching symbols that become single connected components need to be solved in the future because these will affect any possible method used for retrieval that assumes that symbols are separated connected components.

Finally, the indexation task has to be modified in the feature to improve the times required for matching. The current index implementation is just a basic storage of all pre computed features, key frames and neighbor graphs, but no special structures that could improve the matching speed were implemented because of time restrictions and also because the current dataset is relatively small. However, in the future these special structures have to be implemented because the time required to execute a query will grow with the dataset if the index is kept the way it is now.

9. References

1. Liwicki, M.; Bunke, H.. *Recognition of Whiteboard Notes: Online, Offline and Combination*, Singapore: World Scientific, 2008. Print.
2. Calic, J.; Izquierdo, E.; "Efficient key-frame extraction and video analysis" *Information Technology: Coding and Computing, 2002. Proceedings. International Conference on*, pp. 28- 33, 8-10 April 2002.
3. Mohanta, P.P.; Saha, S.K.; Chanda, B.; "A Novel Key-Frame Detection Technique Using Statistical Run Test and Majority Voting," *Computer Vision, Graphics & Image Processing, 2008. ICVGIP '08. Sixth Indian Conference on*, pp.244-250, 16-19 Dec. 2008.
4. Zhao Guang-sheng; "A Novel Approach for Shot Boundary Detection and Key Frames Extraction" *MultiMedia and Information Technology, 2008. MMIT '08. International Conference on*, pp.221-224, 30-31 Dec. 2008.
5. Min, H.; Huazhong, S.; Jing, J.; "An algorithm of key-frame extraction based on adaptive threshold detection of multi-features" *Test and Measurement, 2009. ICTM '09. International Conference on*, vol.1, pp.149-152, 5-6 Dec. 2009
6. Adcock, J.; Cooper, M.; Denoue, L; Pirsiavash, H.; Rowe, L; "TalkMiner: a lecture webcast search engine" *Proceedings of the international conference on Multimedia (MM '10).* ACM, New York, NY, USA, pp. 241-250.
7. Golovchinsky, G; Carter, S; Biehl, J; "Beyond the Drawing Board: Toward More Effective Use of Whiteboard Content". arXiv preprint arXiv:0911.0039. 2009. http://arxiv.org/ftp/arxiv/papers/0911/0911.0039.pdf
8. Parparita, M.; Rusinkiewicz, S.; "Thor: Efficient whiteboard capture and indexing," Princeton University, Tech. Rep., 2004.
9. He, LW.; Liu, Z; Zhang, Z.; "Why take notes? Use the whiteboard capture system," Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on, vol.5, pp. 776-785, 6-10 April 2003.
10. Arjunan, R.V.; Kumar, V.V.; "Image Classification in CBIR Systems with Color Histogram Features" *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*, pp.593-595, 27-28 Oct. 2009.
11. Chiu, CY; Lin, HC; Yang, SN; "A fuzzy logic CBIR system" *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, vol.2, pp. 1171- 1176, 25-28 May 2003.
12. Lowe, DG.; "Distinctive Image Features from Scale-Invariant Keypoints" International Journal of Computer Vision, Volume 60, Issue 2, pp. 91-110, Nov. 2004.
13. Wangming, X; Jin, W; Xinhai, L; Lei, Z; Gang, S; "Application of Image SIFT Features to the Context of CBIR" *Computer Science and Software Engineering, 2008 International Conference on*, vol.4, pp.552-555, 12-14 Dec. 2008.
14. Carson, C.; Belongie, S.; Greenspan, H.; Malik, J.; "Region-based image querying" *Content-Based Access of Image and Video Libraries, 1997. Proceedings. IEEE Workshop on*, pp.42-49, 20-20 June 1997.

15. Jin, C.; Yang, C; “Integrating hierarchical feature selection and classifier training for multi-label image annotation” *Research and development in Information Retrieval*, Proceedings of the 34th international ACM SIGIR conference on, (SIGIR '11). 2011. ACM, New York, NY, USA, pp. 515-524.
16. Datta, R; Joshi, D; Li, J; Wang JZ; “Image retrieval: Ideas, influences, and trends of the new age”. ACM Computing Surveys. Vol. 40, Issue 2, Article 5. 60 pages. May 2008.
17. Cesarini, F.; Lastri, M.; Marinai, S.; Soda, G.; "Encoding of modified X-Y trees for document classification," *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference o*n, pp.1131-1136, 2001
18. Gordo, A.; Valveny, E.; "A Rotation Invariant Page Layout Descriptor for Document Classification and Retrieval," *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp.481-485, 26-29 July 2009.
19. Barbu, E; Héroux, P; Adam, S; Trupin, Ér; “Using Bags of Symbols for Automatic Indexing of Graphical Document Image Databases,” *Graphics Recognition. Ten Years Review and Future Perspectives*. pp 195-205, 2006.
20. Hu, J; Kashi, R.; Wilfong, G.; "Document image layout comparison and classification" Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on, pp.285-288, 20-22 Sep 1999.
21. Van Beusekom, J.; Keysers, D.; Shafait, F.; Breuel, T.M.; "Distance measures for layout-based document image retrieval," Document Image Analysis for Libraries, 2006. DIAL '06. Second International Conference on, pp.11 pp.-242, 27-28 April 2006.
22. Rath, T.M.; Manmatha, R.; "Word image matching using dynamic time warping," *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol.2, pp. 521-527, 18-20 June 2003.
23. Li, L; Lu, S; Tan, CL; "A Fast Keyword-Spotting Technique," Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on, vol.1, pp.68-72, 23-26 Sept. 2007.
24. Lu, S; Li, L; Chew Lim Tan; "Document Image Retrieval through Word Shape Coding," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.30, no.11, pp.1913-1918, Nov. 2008.
25. Marinai, S; Miotti, B; Soda, G; “Digital Libraries and Document Image Retrieval Techniques: A Survey,” Learning Structure and Schemas from Documents, pp. 181-204, 2011.
26. Kacem, A.; Belaïd, A.; Ben Ahmed, M.; “Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context”, *International Journal on Document Analysis and Recognition*, vol. 4, issue 2, pp. 97-108, Dec. 2001.
27. Drake, D.M.; Baird, H.S.; "Distinguishing mathematics notation from English text using computational geometry," *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, vol. 2, pp.1270-1274, Aug. 31 2005-Sept. 1 2005.
28. Garain, U.; "Identification of Mathematical Expressions in Document Images," *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp.1340-1344, 26-29 July 2009
29. Ha, J; Haralick, R.M.; Phillips, I.T.; "Understanding mathematical expressions from document images," *Document Analysis and Recognition, 1995, Proceedings of the Third International Conference on*, vol.2, pp.956-959, 14-16 Aug 1995.

30. Ashida, K.; Okamoto, M.; Imai, H.; Nakatsuka, T.; "Performance evaluation of a mathematical formula recognition system with a large scale of printed formula images," *Document Image Analysis for Libraries, 2006. DIAL '06. Second International Conference on*, pp. 12, 27-28 April 2006.
31. Zanibbi, R.; Yu, L.; "Math Spotting: Retrieving Math in Technical Documents Using Handwritten Query Images," *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp.446-451, 18-21 Sept. 2011.
32. Zanibbi, R.; Blostein, D.; "Recognition and Retrieval of Mathematical Expressions", *International Journal on Document Analysis and Recognition (IJDAR)*, Vol. 15, Issue 4, pp. 331-357, Dec. 2012.
33. Ouyang, T.Y.; Davis, R.; "Recognition of Hand Drawn Chemical Diagrams", Proceedings of AAAI, pp. 846-851, 2007.
34. Jiang, Y.; Tian, F.; Zhang, X.; Dai, G.; Wang, H.; "Understanding, Manipulating and Searching Hand-Drawn Concept Maps", *ACM Transactions on Intelligent Systems and Technology*, Vol. 3, No. 1, Article 11. 21 pages, Oct. 2011.
35. Sciascio, E.D.; Donini, F.M.; Mongiello, M.; "Spatial layout representation for query-by-sketch content-based image retrieval", *Pattern Recognition Letters*, Vol. 23, Issue 13, pp. 1599-1612, Nov. 2002.
36. Leung, W.H.; Chen, T; "Hierarchical matching for retrieval of hand-drawn sketches," *Multimedia and Expo, 2003*. ICME'03. Proceedings. 2003 International Conference on, Vol. 2, pp. II-29. July 2003.
37. Leung, H.; "Representations, Feature Extraction, Matching and Relevance Feedback for Sketch Retrieval". Ph.D. Dissertation, Carnegie Mellon University, June 2003.
38. Kuhn, H. W., "The Hungarian Method for the Assignment Problem", *Naval Research Logistics Quarterly*, Vol. 2, pp. 83-97. 1955.
39. Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M.; "A (sub)graph isomorphism algorithm for matching large graphs," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.26, no.10, pp.1367-1372, Oct. 2004.
40. Luqman, M.M.; Ramel, J.; Llados, J.; Brouard, T., "Subgraph Spotting through Explicit Graph Embedding: An Application to Content Spotting in Graphic Document Images," *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp.870-874, 18-21 Sept. 2011.
41. Luqman, M. M; Ramel, J.Y; Lladós, J; Brouard, T.; "Fuzzy multilevel graph embedding," *Pattern Recognition*, Vol. 46, Issue 2, pp. 551-565. Feb. 2013.
42. Fonseca, M.J.; Ferreira, A.; Jorge, J.A.; "Sketch-based retrieval of complex drawings using hierarchical topology and geometry," *Computer-Aided Design*, Vol. 41, Issue 12, pp. 1067-1081, Dec. 2009.
43. Sousa, P.; Fonseca, M.J; "Sketch-based retrieval of drawings using spatial proximity," *Journal of Visual Languages & Computing*, Vol. 21, Issue 2, pp. 69-80, April 2010.
44. Fonseca, M.J; Ferreira, A.; Jorge, J.A.; "Sketch-based Retrieval of Vector Drawings," in *Sketch-based Interfaces and Modeling*, London, United Kingdom: Springer, 2011, ch. 7, pp. 181–201.

45. Fonseca, J. M; Jorge, A.J.; “Towards Content-based retrieval of technical drawings through high-dimensional indexing”, *Procedings of the 1st Ibero-American Symposium in Computer Graphics (SIACG'02),* Guimaraes, Portugal, pp. 263-270, July 2002.
46. Liang, S.; Sun, Z.X.; Li, B.; Feng, G.H; "Effective sketch retrieval based on its contents," *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol.9, pp. 5266-5272, 18-21 Aug. 2005.
47. Liang, S.; Sun, Z.X.; Li, B.; "Sketch retrieval based on spatial relations," *Computer Graphics, Imaging and Vision: New Trends, 2005. International Conference on*, pp.24-29, 26-29 July 2005.
48. Liang, S; Sun, Z.X.; “Sketch retrieval and relevance feedback with biased SVM classification,” *Pattern Recognition Letters*, Vol. 29, Issue 12, pp. 1733-1741. Sept. 2008.
49. Cao, Y.; Wang, C.; Zhang, L.; Zhang, L.; "Edgel index for large-scale sketch-based image search," *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 761,768, 20-25 June 2011.
50. Liwicki, M.; Bunke, H.; “Feature selection for HMM and BLSTM based handwriting recognition of whiteboard notes”, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 5, pp. 907–923, 2009.
51. Eikvil, L.; “Optical Character Recognition”. Norwegian Computing Center, Oslo, Norway. 1993.
52. Yang, M.; Kpalma, K.; Ronsin, J.; “A survey of shape feature extraction techniques,” *Pattern Recognition*, pp. 43-90, 2008.
53. Szeliski, R.;. “Image alignment and stitching: a tutorial”, *Foundations and Trends in Computer Graphics*. Vol. 2, Issue 1, pp. 1-104. January 2006
54. Bay, H.; Tuytelaars, T.; Van Gool, L.; “SURF: Speeded Up Robust Features”, 9th European Conference on Computer vision, pp. 404-417, may 7-13, 2006.
55. Fischler, M; Bolles, R; “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. *Communications of the ACM*. Vol. 24, Issue 6, pp. 381-395, June 1981.