# Terms

Computer how-to documents

Computer help resources

Computer support resources

Computer knowledge base

Computer knowhow resources

Online technical documents

Technical resources

Resources? Documents? Articles? Webpages?

Online technical resources??

Online computer instructions

Online visual instructions for computers

# Needs for Research

Computer literacy is an important issue. Traditional methods for gaining literacy n computers are taking courses and reading books. This is good for acquiring general computer literacy. There are also many specific computer knowledge. Users can visit the official site of the software vendor or consult the official manual or read the built-in help. But the amount of information is limited. That’s why there’s a market for computer books. Reading the books is hard. It is restricted to the book on the user’s bookshelf. Books can be expensive.

Fortunately, there are tons of such knowhow knowledge on the web.

That’s why people look at the Internet. People no longer need go to bookstores. People gain knowledge from the Web. Some knowledge is general and can be applied over and over. The easy to learn and remember. Some is transient and disposable, only done it once. The goal is to follow the steps accurately.

However, search engine has the largest coverage. It is too general. It’s not suitable for computer knowledge. There is no specialized system for accessing and searching this knowledge base. People are aware of this problem. So they built specialized search engines built for special domains, such as news to show times and images, products to show prices and images, restaurants to show locations and ratings. These were mined from unstructured heterogeneous data.

There are specific needs for each stage of the knowledge life cycle.

**N1. Need to aggregate online visual computer instructions (knowledge aggregation).** Many technical resources already exist. They reside online. They are scattered all over. But there has not been any systematic effort to collect and index them. As a result, looking for the technical resources can be a frustrating experience. Users need to go to many different sites. We need a centralized index. This is beneficial. Such as an index for all the news articles. Users only need to go to one place to search for all the related news reported on different new sites. We need to be able to discover and identify all these different types of technical resources. This can be challenging because these articles take on various forms. They can be about various operating systems, Windows, Mac OS, Linux. They can be called guides, manuals, walkthroughs, tutorials, or how-to. They can be written for novice and experts. Some have only text. Some have screenshot images. Some are videos. The algorithm needs to recognize all of them while at the same time being able to avoid including articles about software but not related to knowhow such as product review and catalogs.

**N2. Need for an indexing scheme tailored to the properties of online computer instructions (indexing).** Once technical resources are collected, we need to index them. Traditionally indexing scheme uses keywords. For documents are broken down into keywords. For videos, transcripts can be used if available, or user supplied metadata tags. An inverted index is constructed based on extracted keywords for efficient retrieval, where each word is a key and stored under each key is a list of resources related to the word. However, this scheme is not optimal. It has been observed that users can follow articles faster when the articles contain screenshots to illustrate each step compared to text-only articles [cite]. Visual information is very crucial and useful. It is conceivable users may prefer articles with screenshots. Since these articles are indexed only by the keywords they contain, there is no way to search them by visual contents. Moreover, for efficient retrieval, we need a way to break images or video frames down into *visual* keywords as well so that we can add them to the inverted index. Thus, there is a need for an optimized indexing scheme that not only considers both textual and visual content but also lends itself to the construction of an inverted index for efficient retrieval.

**N3. Need for an intuitive input method for specifying query terms (querying).** Current input method for specifying queries is difficult. It is hard to come up with the right keywords. To identify the context, there are too many things need to be specified. Users need to indicate the operating system, the name of the application, the name of the window, and the topic, what the users want to know. The result is a long list of keywords, which can be ambiguous. The context and topic are indistinguishable from the search terms. From usability point of view, it is time consuming to type all the keywords. Thus, there is a need for a more intuitive method to specifying queries. It is even crazier for searching video instructions.

**N4. Need for a ranking scheme considering a rich set of features (ranking).** If we allow query to consist of both an image and a set of keywords, we need a new ranking scheme. A typical search engine ranks retrieved Web pages based on how closely keywords are matched to the query words. A content-based image retrieval system ranks retrieved results by similarity to the query image. Neither is ideal. Simply ranking articles by images may result in articles at the top containing no useful text. Simply ranking articles by text may result in informative articles but on the wrong computer application. A new ranking scheme that consider both text and visual relevancy to the queries needs to be developed to guarantee the top results are both textually and visually relevant.

**N5. Need for a scheme to display excerpts to help users judge relevancy quicker (previewing).** Results of Web search are typically presented to users as a list of links accompanied by short excerpts. As users browse through the list, they rely on these excerpts to decide which links are likely to be relevant with respect to their search objectives. Thus, it is critical that an excerpt should adequately represent the content of a page and should be obvious about its relevancy to the search objective. Thus, the keywords are highlighted. However, for computer instructions, this is inadequate; excerpts are often in pure text. An excerpt may contain all the keywords, but may be about the wrong application. Video excerpt not good. Pure image exceprt not good.

**N6. Need for a more effective, convenient, interactive way to follow knowhow articles (consuming).**

After retrieving an instructional article, users will follow the steps outlined in the instructions while performing each interaction taught by each step. Usually, the instruction is displayed in a separate Web browser. Following the instruction can be tedious exercise. They need to keep switching back and forth between the application and the browser. It is known synchronized viewing is useful for programming. This feature is not available for ordinary computer users. It is easy to lose track. Sometimes users may already be at a particular step; they may wish to search within the document to the relevant step. Sometimes the page may contain several instructions for several things. The users may wish to jump to only that part. It is even harder if the instruction is a screencast. Users need to pause the video player periodically. Users can only go through the video sequentially. Some users may remember the first couple steps. They want to refind the instruction. However, they are forced to watch the instructional video from the beginning. They cannot jump directly to the relevant segment.

**N7. Need for an open platform for creating contextually accessible technical resources (contributing).** When creating a technical resource, it is often desirable to link the resource directly to the application window it covers. Many developers have done this. They add hooks to GUI components. Some resources are short phrases such as tooltips that can be accessed by mouse-over. Some resources are popup messages that can be opened by clicking on a question mark. Some resources are web pages that can be opened in a browser by pressing a hotkey such as F1. However, this linking operation is a privileged operation that can be performed only by developers or special people with access to source code or special API. However, many ordinary people who wish to create technical resources but they can never link these resources to the application. They can put them online and hope the search engine will find them and put them in an index. To access the resource, users need to use the search engine to be link to the resource indirectly. Hopefully users can specify the right keywords. For example, a computer instructor may wish to create a tutorial for the network preferences window. But unless the instructor has access to the source code or a special API, it is not possible to link the tutorial to the application window directly. Thus, there is a need for an open platform for creators to easily link online technical resources , link them to applications so that users can access them directly from the application.

# Goals and Objectives

The overarching goal of the proposed project is to address the needs identified in all stages of the knowledge life cycle as described above.

Develop an algorithm for detecting articles containing computer knowhow.

Users should be able to ….

# Proposed Methods

Overview. We propose a multi-modal approach. We use images and text.

## Aggregation

Our first research goal is to develop a method to use existing search engines as bootstrap to college online visual computer instructions to meet the **need for a centralized knowledge base (N1)**. We plan to reach this goal by forming three objectives and meeting them one by one.

### Objective 1: Collect candidate pages using existing search engines

We will specifically target those pages containing computer screenshots as likely candidates. We will use an image search engine and a reverse image search engine to find these pages.

**Image Search.** An image search engine such as Google Image Search takes a set of words as search terms and returns a list of links to online images discovered on pages containing these words. To use an image search engine to collect candidate pages, we will manually compile a keyword list consisting of words sampled from the title bars of various computer application windows. Some examples of these keywords can be properties, preferences, option, settings, wizard, installation, network, sound, and keyboard. We will append to the list additional keywords that are commonly found in computer instructions such as tutorial, guide, and manual. Then, we will systematically submit different combinations of the keywords in the list to an image search engine. The rationale is that these keywords are likely to retrieve screenshot images of a variety of application windows. The pages containing these screenshots are likely to provide useful computer instructions related to these application windows, thus our candidate pages.

**Reverse Image Search.** A reverse image search engine like TinEye takes an image as the query and returns a list of links to online images visually similar to the query image. To use a reverse image search engine to collect candidate pages, we will create a list of query images by manually capturing the screenshots of a wide variety of common application windows across popular OS platforms (XP, Vista, and MacOS). We will submit each screenshot as the query to a reverse image search engine to retrieve a list of similar screenshot images and links to their source pages. These pages will be the candidate pages.

### Objective 2: Develop an algorithm to analyze the content of the candidate pages and identify those that are computer instructions.

While an image search engine will retrieve candidate pages containing a wide variety of textually relevant images, there is no guarantee every image is a computer screenshot. On the other hand, a reverse image search engine can almost guarantee every retrieved candidate page must contain a visually relevant computer screenshot. But in terms of the content, there is no guarantee every page is about computer instructions. Therefore we will develop an algorithm to classify the candidate pages into true the false positives. Given the text and the image of a candidate page, the algorithm must determine whether the image is a computer screenshot and whether the text is about computer. To determine whether the image is a screenshot, we will consider visual features such as the size, shape, color, and texture of the image and the presence of salient visual components such as title bars, buttons, and checkboxes. To determine whether the content is about computer instructions, we will manually label a large sample of candidate images as positive and negative examples. We will use these training examples to train a binary classifier such as SVM. In addition to image and text features, we will also consider their structural relationship. For example, if a page exhibits the pattern of short sentences interspersed with screenshots, it can be a sign that the page is some kind of step-by-step instructions.

### Objective 3: Crawl the sites found to host visual computer instructions to systematically collect other computer instructions.

Then we will use spider to systematically visit all the links in these seed sites. So we can collect other technical resources that are likely to be hosted on the sites too.

Also talk about where to collect videos.

## Indexing

After constructing a large corpus of visual computer instructions, our next research goal is to develop a scheme to index these resources to support fast retrieval to meet the **need for a specialized indexing scheme (N2)**. Also, this indexing scheme needs to take advantage of multiple modalities. There are three candidates for inverted index. We propose to index knowledge by text and images. Specialized, multi-modal indexing scheme. Compared to keyword only and text only. Stemmed words. Cluster centers. OCR stemmed words. Top 3 candidates. Features computed offline. Features are independent of queries.

We will consider three types of primary key features.

Our prototype system indexes screenshots extracted from a wide variety of resources such as online tutorials, official documentation, and computer books. The system represents each screenshot using three different types of features (Figure 2). First, we use the text surrounding it in the source document, which is a typical approach taken by current keyword-based image search engines.

Second, we use visual features. Recent advances in computer vision have demonstrated the effectiveness of representing an image as a set of visual words [18]. A visual word is a vector of values computed to describe the visual properties of a small patch in an image. Patches are typically sampled from salient image locations such as corners that can be reliably detected in despite of variations in scale, translation, brightness, and rotation. We use the SIFT feature descriptor [11] to compute visual words from salient elliptical patches (Figure 2.3) detected by the MSER detector [12]. Hamming Embedding. Spatial Pyramid Matching.

Screenshot images represented as visual words can be indexed and searched efficiently using an inverted index that contains an entry for each distinct visual word. To index an image, we extract visual words and for each word add the image ID to the corresponding entry. To query with another image, we also extract visual words and for each word retrieve from the corresponding entry the IDs of the images previously indexed under this word. Then, we find the IDs retrieved the most number of times and return the corresponding images as the top matches.

Third, since GUI elements often contain text, we can index their screenshots based on embedded text extracted by optical character recognition (OCR). To improve robustness to OCR errors, instead of using raw strings extracted by OCR,

we compute 3-grams from the characters in these strings. For example, the word system might be incorrectly recognized as systen. But when represented as a set of 3-grams over characters, these two terms are {sys, yst, ste, tem} and {sys, yst, ste, ten} respectively, which results in a 75% match, rather than a complete mismatch. We consider only letters, numbers and common punctuation, which together

define a space of 50,000 unique 3-grams. We treat each unique 3-gram as a visual word and include it in the same index structure used for visual features

Online screenshot poses several challenges to visual similarity matching. While certain challenges are absent such as lighting variation and rotation, there are other unique challenges. Since there are sime challenges we do not need to deal, we can trim down the algorithm to make it leaner for increased efficency. Talk about challenges. Problems with variations. For screenshots, translations not a big problem. But scale is problem, theme variation is problem. For videos, translation is often problem. Full screen capture. But the query can be restrictd to cropped image. Animations. There’s no text. The text can be derived from the trancript because they tend to do voice over. We use automatic speech recognition such as the one used by Google voice. We use words learned from similar screenshots in the web pages to set the prior for picking words and to disambiguiate words, to improve speech recognition results.

Detect types. Classifying into categories.

## Querying

The goal is to develop a query interface to meet the **need for a simpler query input method (N3)**. Queries have two components: context and topic. Users have to enter many keywords to describe the application and to describe the desired piece of knowledge. Our approach is to allow users to use images of the application as a search term to eliminate the need to identify the application with keywords. It provides a clear separation of context and topic

Screenshot queries can be specified in two ways. First, users can run a cross-platform, Java-based, client interface we developed to capture the screenshot of a selected window or an arbitrary screen region. The client interface will submit the screenshot as the image query to our search engine and display the search results in the default web browser. Alternatively, users can use any existing image capture utilities such as the snipping tool on Windows Vista or the Command-Shift+4 hotkey on Mac OS to capture screenshots. Users can use a Web interface to submit the screenshots to our search engine and view the results directly in a Web browser. . Use Java to provide cross-platform applicability. Allow users to take multiple screenshots as query. User Interface for Searching Screenshots Sikuli Search allows a user to select a region of interest on the screen, submit the image in the region as a query to the search engine, and browse the search results. To specify the region of interest, a user presses a hot-key to switch to Sikuli Search mode and begins to drag out a rubber-band rectangle around it (Figure 1). Users do not need to fit the rectangle perfectly around a GUI element since our screenshot representation scheme allows inexact match. The image matching method is robust to some amount of background. After the rectangle is drawn, a search button appears next to it, which submits the image in the rectangle as a query to the search engine and opens a web browser to display the results. We will also support the selection of the GUI component. This feature is common in current screen capture utility.

Keywords queries are optional and can be specified in three ways. They can be entered together with the screenshot queries on both the client and Web interfaces. Also, as the users are browsing the results, they can enter keywords to filter and/or refine the results.

Not sure which combination is the best. We will research on this. Keyword first, or image first, or no ordering. How many keywords can be saved? What keywords are redundant or replaced by images. We will design the initial beta interface with the flexibility for users to do all of them and study the behaviors.

## Ranking

The goal is to consider ranking to meet the **need for a more relevant ranking method (N4**). A query can result in many possible matches, may retrieve several matches from the index. These matches need to be ranked. The goal is that the top ranked ones are those desired by the users. Indexing is done independent of queries. Ranking is done with respect to the query. What are the technical challenges? How can we combine multiple indexes? We have three primary key features: words, visual words, and ocr words. There are other features not primary keys. There are also other features that can influence the relevance. Some features are query dependent. Some are independent. All primary key features are query independent. What are they? What are their relative importance?

List a set of features we will consider:

Some of these features are common in other information organization problem. Provide citations that inspire these features.

Some features are unique in our domain.

* Number of screenshots
* Number of words per screenshot
* Number of action words
* Number of steps
* Number of instructions hosted on the same site
* Image size ratio
* Word distance between relevant keywords and screenshots
* Length of video
* Length of relevant segment
* Number of segments containing relevant frames
* Time distance between relevant keywords and frames

Research questions are, what features are important. How can we learn them? We propose two ways to collect training data. We will use AMT collected preference judgment of unranked results. And apply RankSVM to learn weights.

When we show the results to AMT workers, all these features need to be visible. However, showing all of them will be overwhelming. Thus, each time we select a random subset of features. Overall, we can still collect a sizable trining data set to do some meaningful learning.

Since not all features are equally important, it is necessary to set the weights appropriately in order to reflect these features' relative importance. In developing the prototype of our system, we set the weights in three stages. Table \ref{tbl:feature\_weights} lists all the features and indicates the stages when the weights are set. First (stage 1), we apply RankSVM to learn feature weights based on the training data collected using Amazon Mechanical Turk. RankSVM was

originally proposed by Joachims \cite{Joachims} to learn how to weight features from a set of subjective ordering constraints inferred from user click-through data to improved ranking. At the development stage, we did not have any click-through data to derive ordering constraints for RankSVM. Thus, we recruited workers from Amazon Mechanical Turk to provide us explicit article ratings from which to infer ordering constraints. We chose five query images known to retrieve a large of

number of articles, but they are ranked only by visual similarity. We shuffled the order of these articles and presented them in a list, Each worker was asked to rate each article in terms of usefulness on a 5-point scale. For quality control, we included two junk results and two duplicate results. To pass the quality test, workers must

successfully mark junk results as useless and give consistent ratings to duplicate results. Then, each pair of ratings provided us with an ordering constraint for training RankSVM. Note that since there was no easy way to ask the workers to provide meaningful keywords, this procedure only applied to the subset of features that do not depend on keywords.

## Previewing

The goal is to design a good presentation scheme to meet the **need for an effective display scheme for previewing results (N5).** Why is previewing important? It enables users to quickly scan through a list of results and identify a subset of those that may be very relevant. The goal is to make it easy for the users to preview the resource and to predict relevancy very quickly. The goal is users can quickly browse the results and evaluate whether each result is relevant. Make a list of design objectives. Users should be able to quickly decide the image matching is correct. Users should be able to also decide the text is relevant. Users should be able to read the relationship between image and text. Users should not read lots of stuff. Users should be able to scan down the list. Users should be able to filter the results.

##### Our approach: Show the matched screenshot’s thumbnail. Extract excerpts from the following locations: title, nearby text…etc. Show tiny version of the screenshot in the context of a paragraph, the sentence before and after the image. Arrange in the top down list. We will provide faceted interface for users to filter results.

For video results, the needs are similar. the user should be make sure the video does contain the desired application. make sure the video also mentions the desired topic. So maybe shows the keyframes before and after where the query image appears. And also shows the transcription text before and after. Also, make the keyword highlighted in the transcription. Provide visualization to show whether a word is mentioned during the segment, or prior to or after the segment.

Another issue is how to display three kinds of results. Should we integrate them in a single ranked list? First, we will put video results in a separate section, like Google. Second we will integrate the results together in the same ranking. Ideally, video result should be coherent with regular results; they can be displayed together in the same list. Maybe not. Google has shown it is useful to clearly separate the video search results. We shall test and compare the two designs. How to present videos and documents in the same page? We can model after current search engine. Typical search engine only shows videos as the whole. We want to show by individual clips. Only the clip within the video.

Top features learned in the previous step will be shown in the preview. We speculate that the title, position of image, and neaerby text, heading are important information. Thus, a prototype preview format scheme can be seen below.

Provide faceted search function.

## Consuming/Internalizing/Receiving/Digesting/Reading/Following/Learning/Learning

## Reading

Research goal is to build tools to meet the **need for a better support for learning (N6)**. The ultimate goal is to help users follow a tutorial and learn something useful.

When user opens a page in the web browser, the page may contain several screenshots. If desired, the users should be able to scroll to the part of the page displaying the relevant screenshot. To do this, we can build a local index of the images, if the page contains a lot of images. If the number of images is small, we can linear scan. The user can stay on the same page. And jump to the next image automatically. The user move to the next screen, the user can search for the corresponding image within the same document.

For video viewing, user should be able to easily synchronize their interactions with viewing. Users should not need to periodically pause the video. Users should be able to seek within the video just like users can search within a text document.

Synchronized viewing. To detect transition, we take a visual approach. We periodically capture the screen. For video player, we detect transition to know when we should automatically pause the video. For the application, we detect transition to know when we should automatically play the video. We do not need to access the API to do this. This approach is universal. To increase efficiency we only need to monitor an area marked by the user. Users can explicitly draw an area to engage synchronize viewing.

Visual seek. To provide random access within a video, we must preprocess the video. The processing can be done offline or online. If a video is not already in the index, users should be able to load the video and compute the index on the fly. If the video is already in the index, a data strcture that can support rapid visual seek should have been computed and stored as meta data.

## Contributing

The goal is to allow anyone to contribute new knowledge that can be linked directly to the applications. This concerns content contributors, or contributors of instructions. Contirubors should be able to easily author new instructions rich with visual illustrations and link the instruction to the application. We took a visual linking approach. Allow content providers to attach arbitrary information to programs by images. Establishing visual links. Submit an url to the system. And it downloads all the image and index the page. People can contribute their knowledge. We plan to explore the use of screenshots as hooks to link applications to technical resources. Annotation systems are common on the web (e.g. WebNotes and Shiftspace), where URLs and HTML page structure provide robust attachment points, but similar systems for the desktop have previously required application support. Using screenshots as queries, we can provide general-purpose GUI element annotation for the desktop, which may be useful for both personal and community contexts. We plan to build an annotation interface allows a user to save screenshots with custom annotations that can be looked up using screenshots. To save a screenshot of a GUI element, the user draws a rectangle around it to capture its screenshot to save in the visual index. The user can create a web page and insert this screenshot. User can insert multiple screenshot as well. The user can upload the page to the repostitory. The screenshot will be analyzed and stored as an image link.

Advantages: easy to kept up to date. Cheap. Open.

Creating image links can bypass the application and system and API layer.

When creating the video, users can just record as is. They do not need to do any annotation. They can then upload the video to our service. Creators can explicitly upload the video to our service for indexing, or simply wait till our spider hits the video. The service will analyze the video and index its frames for quick reverse lookup. A link will be added from the current screen to frames in the video. Or link to online responsitory. User can have a program that watches the entire screen. It will check again the database and highlight those components that have screenshots in the database. User can click on the component to follow the link to read pages about the component. User can also select a region so that the link lookup will be restricted to that region for efficiency reason.

# Evaluation

Develop a scheme that uses images to create context and evaluate against a sizable dataset and show statistical significance more relevant results than start-of-the-art methods, when evaluated by human users. First page result. Above the fold result. Recall and precision. Use inverted index for fast retrieval. Offline processing.