• Information Integration and Informatics (III)

Recent years have seen massive growth in the scale, diversity, and complexity of data. Moreover, the data are often used in unanticipated and new ways that frequently require repurposing, transforming, and/or integrating multiple, uncoordinated, and sometimes variously restricted data sources over which data users have no control. The abundance and heterogeneity of data and data sources have created increasing demands on and opportunities for information technologies.

The Information Integration and Informatics (III) program focuses on the processes and technologies involved in creating, managing, visualizing, and understanding diverse digital content in circumstances ranging from individuals through groups, organizations, and societies, and from individual devices to globally-distributed systems. Further, data are only part of a “knowledge life cycle” that progresses from data through knowledge and insight and, ultimately, to action. III funds innovative information technology research that can transform **all stages of the knowledge life cycle**.

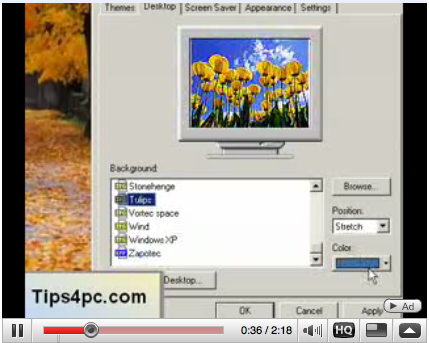
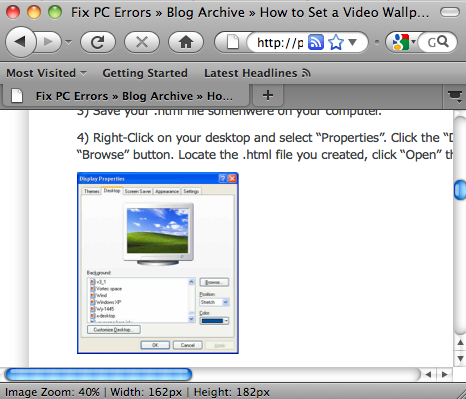
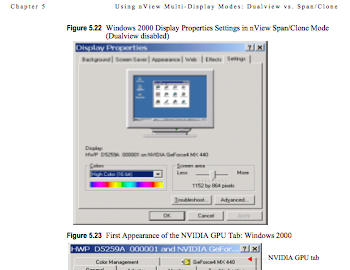
III-funded projects are expected to lead to advances that are driven by specific i**nformation-technology challenges**. Projects directed mainly at data-collection building and use, that apply existing data technologies to (perhaps) novel data sets, or that propose other activities with limited computing and information technology research potential are not appropriate for this program. III-supported activities can range from theoretical investigations to projects grounded in **multi-disciplinary collaborations** where data are central to the III-area research. In the case of multi-disciplinary projects proposers should explain the **utility of the proposed work to the application domain** and demonstrate **expertise in that domain** among the project participants. Regardless of research modality, proposals should make clear what computing and information technology challenges are being addressed and how the effectiveness of the work will be assessed

# III: Small: an integrated system for indexing, searching, and browsing online screenshot instructions using text and images

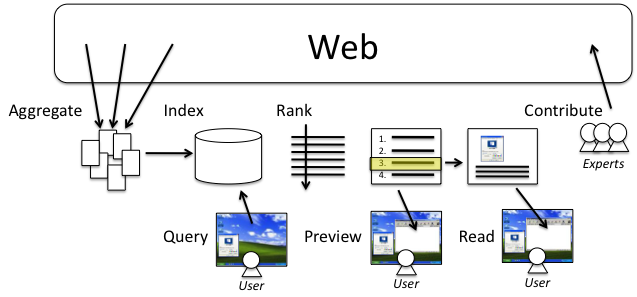
Learning never ceases. In the increasingly digitalized world, we frequently encounter an unfamiliar computer task we need to learn how to perform. For example, we may want to learn how to set up a laptop to access a virtual private network or how to configure a printer to print in the duplex mode. Sometimes it may be the first time we need to perform the task. But often we may have done the same task a while ago and simply have trouble remembering the details. In both cases, we would like to find relevant step-by-step instructions we can read and learn how to perform the task.

The most effective kinds of instructions are those with screenshots. Harrison [Harrison] found that users learned better following instructions illustrated by screenshots than reading textual-only instructions. However, screenshot instructions are relatively rare in user manuals or books because they occupy more space and cost more to print.

Fortunately, there is abundance of screenshot instructions on the.Unconstrained by high printing cost, web pages giving computer instructions can include as many screenshots as possible to maximize their usefulness. There is also a growing trend to create screenshot instructions as videos thanks to the popularity of video sharing sites such as YouTube. As a result, new screenshot instructions are constantly being published by a wide variety of people. Software developers regularly publish screenshot instructions about their software products to complement builtin documentation (e.g., support.microsoft.com). Organizations such as companies and schools often post screenshot instructions to teach their members how to perform important tasks ranging from installing anti-virus software to configuring email clients. Perhaps the most prolific creators of screenshot instructions are altruistic expert volunteers who frequently offer tutorials and share tips about software applications on forums or blogs for the benefits of all. It is estimated that screenshot instructions available online are numbered in tens of millions [cite]. These screenshot instructions represent a tremendous amount of collective knowledge about computers and create research opportunities on how to harness such knowledge effectively.



Examples of screenshot instructions in a printed manual (left), a webpage (middle) and a video (right)



The **knowledge life cycle** of online screenshot instructions consists of seven stages. Over time, many experts have **contributed** a large amount of screenshot instructions in the form of web pages and videos. A search engine can **aggregate** these screenshot instructions and **index** them to make them searchable. A user may wish to learn how to perform a particular task and can **query** the search engine with appropriate search terms. The search engine can **rank** the results based on their relevancy to the search terms. The user can **preview** the results and choose the one that seems the most relevant. Finally, the user can **view** the instruction to learn how to perform the task.

# Needs for Research

However, current general-purpose search engines provide limited support throughout the knowledge life cycle of online screenshot instructions.There are strong needs for research on both the processes and technologies to improve all parts of the knowledge life cycle.

* **Need for an algorithm to detect and aggregate online screenshot instructions (aggregation)** There has not been a systematic effort to collect and index screenshot instructions scattered all over the Web. As a result, finding relevant screenshot instructions can be a frustrating experience for many users. Users may need to visit several technical support sites and browse through the directory before they can find what they want. Even if they use a search engine, they may retrieve many useless pages that do not offer any instruction and need to painstakingly sift through them. Therefore, a centralized, searchable knowledge repository of screenshot instructions can be very beneficial to many users. The construction of such repository poses several interesting research questions. How can we identify sites with high-density of screenshot instructions? How can we automatically determine whether a webpage or a video clip contains screenshot instructions? How can we detect if an image or a video frame is a screenshot of a computer application? There is a need for research to answer these questions.
* **Need for an indexing scheme using text and image features as primary keys (indexing).** Once we have aggregated a large, unorganized collection of screenshot instructions, we need to build an index so that these instructions can be efficiently searched. To build an index, it is critical to choose a suitable feature as the primary key. Many search applications use words as the primary key. However, since screenshot instructions contain both images and text, using only words as primary keys will only capture the content partially. Therefore, there is a need for research to identify the types of visual features that can serve as the additional key to index screenshot instructions effectively.
* **Need for UI innovation to reduce users’ querying efforts (querying).** Keywords can be difficult to use for many users to search screenshot instructions. Users must come up with the right keywords to indicate both the context (which screenshot) and topic (what instruction) they desire to find in a webpage or a video. They may need several words to indicate the type of operating system, the name of the application, the name of the window, and possibly more. They may need even more words to describe the desired topic. The result is a long list of keywords, which is not only mentally and physically tedious but also prone to ambiguity (e.g., does the word “setup” refer to a program or an action?). Therefore, there is a need for research to find out whether allowing users to use screenshot as queries can reduce to reduce querying efforts.
* **Need for a ranking scheme that combines textual and visual features (ranking).** Most search engines rank results based primarily on the textual relevancy to a keyword query. On the other hand, some content-based image retrieval systems [cite] are able to rank results based primarily on visual relevancy to an image query. However, neither ranking scheme is suitable for screenshot instructions. Ranking by textual relevancy may return instructions with the wrong screenshots. Ranking by visual relevancy may find the right screenshots but on a wrong topic or offering no instruction. Therefore, there is a need for research to develop a new ranking scheme that respect both textual and visual relevancy so that users can find among the top results useful instructions that are visually and textually relevant.
* **Need for a method to generate excerpts to convey textual and visual relevancy (previewing).** A typical search result returned by a search engine is 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 and worth exploring. However, current excerpts are text-oriented and provide no evidence regarding the visual relevancy of the search results. It can be difficult for users to judge whether some results are really relevant. Therefore, there is a need for research to develop a method to extract relevant textual and visual snippets and to generate excerpts easier for users to judge relevancy.
* **Need for an effective support for viewing screenshot instructions (viewing).** Current web browsers and video players do not provide adequate support for reading screenshot instructions. As users try to follow the steps outlined in an instruction, they often need to switch back and forth between the application and the viewer periodically, which can be a tedious exercise. The experience can be even more frustrating for videos because of the constant need to pause and play the video between steps. Moreover, as users perform an action and see the next screen, there is no automatic way to scroll to the part of the page or fast-forward to the segment in the video corresponding to that screen. Because of these deficiencies, even if we can build a search engine to retrieve relevant screenshot instructions for users, the benefits cannot be fully realized. Therefore, there is a need for research to provide effective supports for viewing screenshot instructions.
* **Need for an accessible method to create context-sensitive instructions (contributing).** When creating an instruction about a program, it is desirable to link the instruction directly to the program. For example, it has been a standard practice for application developers to add a hook to each screen of the application to provide context-sensitive help. Users can press a hotkey such as F1 at any screen to trigger the hook to link to the documentation about the screen. However, the ability to establish context-sensitive links requires access to the source code or some special API. Many people either do not have such access or do not possess the necessary programming expertise. As a result, most screenshot instructions created by third-party users are published on the Web separated from the applications that do not allow context-sensitive access from the applications. Therefore, there is a need for research on whether screenshots that can be captured by ordinary people can serve as context-sensitive hooks to link applications to relevant online instructions.

# Proposed Research Activities

## I. Aggregation

Our first research activity is to figure out a way to aggregate online screenshot instructions. We plan to proceed in three steps. First, we will explore the use of existing search engines as bootstraps to collect a large pool of candidate web pages or videos that are likely to contain screenshot instructions. Then, we will develop an algorithm to determine whether each candidate is indeed a screenshot instruction by analyzing its content. Finally, given a set of screenshot instructions already found, we will systematically crawl their source sites in order to collect other screenshot instructions hosted on the same sites.

### Collecting candidates using existing search engines

We will use an image search engine, a video search engine, and a reverse-image search engine to collect candidates.

* **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 candidates.
* **Video Search** A video search engine in principle works like an image search engine. It takes keywords as search terms and retrieve links to online videos that may be titled or tagged with related keywords. We will use the same set of keywords as above to retrieve candidates of videos offering computer instructions.
* **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.

### Detecting screenshot instructions among the candidates

A significant number of candidates obtained using existing search engines may not be screenshot instructions. While an image or video search engines will retrieve candidates containing textually relevant images, there is no guarantee every image is a computer screenshot. For example, the word “display properties” can retrieve screenshots of the display properties window as well as images of properties displayed by realtors. On the other hand, given screenshots as queries, a reverse image search engine can almost guarantee every retrieved candidate must contain a screenshot. But there is no guarantee the screenshot is part of an instruction; it can be from a software catalog.

Therefore, we need to filter the candidates and keep only those that are screenshot instructions. We need to develop an algorithm that can analyze both the text and image content of each candidate and determine whether the text is about instructions and whether the image is a computer screenshot respectively. To determine whether the text is about computer instructions, we will consider word frequency features. 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. We will manually label a large sample of candidates as training 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.

### Crawling systematically to collect all other screenshot instructions

The previous two steps will give as a large initial pool of screenshot instructions. We can consider the sites containing these screenshot instructions as seed sites. We will use a spider to systematically visit all the links in these seed sites, hopefully to discover other screenshot instructions hosted on the same sites.

## II. Indexing

Our next research activity is to develop an indexing scheme for screenshot instructions that supports fast retrieval and allows multimodal primary keys.

### Building an inverted index for fast retrieval

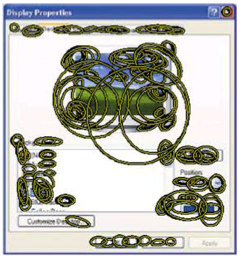
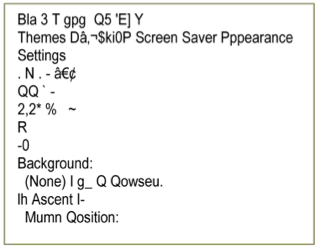
An inverted index is the standard indexing scheme to support fast retrieval. Given a set of documents and their features, a forward index takes a document ID and retrieves a list of features owned by that document, whereas an inverted index takes a feature and retrieves the IDs of the documents that possess the feature. Given a feature as a query, finding a document with that feature from a forward index requires iteratively retrieving each document until one with that feature is discovered. The query time is linear to the number of documents in a database, which is undesirable if the database is large. In contrast, given the query feature, an inverted index can retrieve the document associated with that feature immediately.

### Computing multimodal primary keys

A feature suitable for primary key must have four properties. It must be discriminative. It must be robust to variations. The number of possible features must be much smaller than the number of entries in the database so that inverted lookup can be much more efficient than forward lookup. Also, given a feature, it is desirable to be able to find the matching primary key very efficiently (sub-linear time).

We will consider four types of features as primary key.

* **Word** The standard choice for the primary key for indexing text documents has been words. Many well-established techniques have been established to use words effectively as primary keys. For example, to make words robust to variations, we can compute stemmed words. To make them discriminative, we can use a stop list to ignore common words. To allow sub-linear time lookup, we can store words in a hashtable.
* **Visual Word** To index computer screenshots, we will need visual features with the same properties as words. Recently, many features have been proposed in the computer vision literature that can provide these properties for large-scale image database applications. One example is the SIFT feature. These features can be extracted from salient regions in images. They are robust to variations in scale and translation, which is important because computer screenshots can be resized or cropped. It is discriminative. There are techniques for sublinear time lookup of similar SIFT feature such as the nearest-neighbor techniques. 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 will use the SIFT feature descriptor [cite] to compute visual words from salient elliptical patches detected by the MSER detector [cite].
* **OCR Word** Since computer screenshots 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. If we consider only letters, numbers and common punctuation, we can define 50,000 unique 3- grams as primary keys.
* **Speech Word** Since videos often have voice commentary, this commentary can be analyzed to extract the words spoken in the commentary. Given the inherent unreliability of speech recognition, we can take an approach similar to one above for OCR words to generate primary keys.

********

Examples of visual words and OCR words extracted from a screenshot

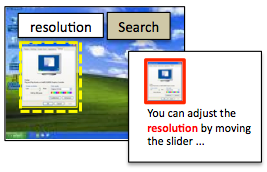
### Exploiting cross-modality redundancy

We will exploit the redundancy across multiple modalities to improve the recognition results of each modality. For example, we can improve speech recognition results by learning the distribution of words on pages containing visually relevant screenshots. There can be high correlation between the words written on pages or spoken in videos about the same screenshot.

## III. Querying

The third research activity is to design a novel interface to reduce users’ querying efforts. Our approach is to support multimodal queries consisting of both screenshots and keywords. We expect this interface to provide the following usability advantages:

* **Minimal input efforts** We will allow users to use application screenshots as query to eliminate the need to enter keywords to describe the application.
* **Good learnability** The interaction style for specifying the region for screen capture will be modeled after that of drawing a rectangle in a typical graphic editor. This interaction style is familiar to many users and will be easy to learn.
* **Application independency** We will obtain the pixel data directly from the screen buffer to represent the target application. This does not require any formal application support.
* **Platform independency** We will implement the interface using Java technology.
* **Consistency** We will make screen capture the primary trigger mode, since it is possible to do for every application.
* **Context-topic separation** We will train the users to use screenshot to specify the context and provide additional keywords to describe the desired topic.



Our proposed query interface will enables user to take a screenshot to indicate the context and enter keywords to specify the desired topic.

## IV. Ranking

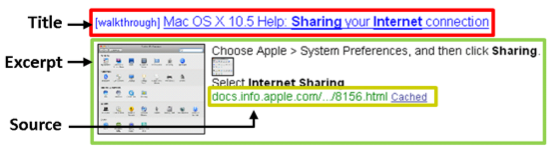
A query can often result in many candidate matches from the index. These candidates need to be ranked. Our next research activity is to develop a ranking scheme that incorporates both visual and textual features. We identify the following features that may be important for ranking:

* **Text features:** the number of matched keywords, the number of action words (e.g., click, open), and the number of steps.
* **Image features:** the similarity value of the matching screenshot, the size ratio between the query and the matching screenshot.
* **Video features:** the total length of the video, the number of matching video segments, and the length of each matching video segment.
* **Multimodal features:** the number of words per screenshot, the number of lines between the matching keyword and the matching screenshot, and the spatial relationship between words and screenshots (i.e., linear or grid).
* **Page features:** the number of words on the page, the number of other screenshots on the page.
* **Site features:** the number of other computer instructions hosted on the site, the popularity of the site.

After extracting a set of features from screenshot instructions, we will learn how to weight them. While we know all these features may play a role in relevancy judgment, we do not know what features are more important than the others. To learn weights, we will apply RankSVM [Joachims]. This learning technique was proposed by Joachims for learning feature weights from a set of subjective ordering constraints inferred from user click-through data to improve ranking. Since we do not have any click-through data initially, we will recruit annotators to provide us with ratings of a set of results. We will also generate simulated queries. For each query, we will retrieve a set of unranked results. We will shuffle their order and present them in a list. Also, we do not know which features will be important. Annotators must be able to view the features in order to judge the relevancy based on the features. However, showing all of them will be overwhelming. Thus, each time we will select a random subset of features to show.

## V. Previewing

After a ranked list of results is produced, the next research activity is to develop a method to generate excerpts for users to preview the results and determine whether they are relevant, and if so, read the full content. To judge relevancy, the three most important things a user may want to check are: (1) is the screenshot is correct, (2) are the keywords matched, and (3) how are the matched screenshot and keywords placed on a page or in a video. To enable the user to check these, we propose the following design for previewing a single result.



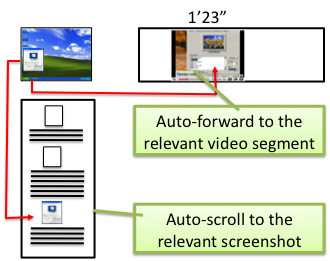
In addition, from the previous research activity, we will have learned how different features would affect the ranking. Features with higher weights should also be visible. For example, if the size of the image turns out to be important to ranking, it should be displayed prominently in the preview as well.

For videos, users also rely on preview to judge the relevancy of a candidate video before actually playing the video. Users may want to make sure the video not only contains the right screenshot but also mentions the desired topic. Thus, we will show the location of the matched screenshot in the video as the evidence of the presence of the right screenshot. We will highlight the occurrences of the topic words in the transcript. Moreover, to relate the matched words and screenshots, we will provide visualization to show whether the words are mentioned during, before, or after the matched screenshot.

## VI. Viewing

The next research activity is to investigate how we can provide better supports for users to view online screenshot instructions. To overcome the limitations of current viewing tools, we propose the following two enhancements:

* **Visual seeking** When users open an instruction page in a web browser or load an instruction video in a player, there may be multiple screenshots. Visual seeking allows users to find the relevant screenshot and advance to the screenshot automatically. To support this feature, we will pre-compact a local searchable index of screenshots for each page. If the number of screenshot is small, this index can be a forward index using linear search. Otherwise, the same inverted index scheme used for the global index will be applied for efficiency.

****

* **Synchronized viewing** When using an application while following an instruction, synchronized viewing allows users to automatically scroll a page or advance a video to the next screenshot as users move to the next application screen. This feature will eliminate the need to switch back and forth between the application and the browser and the need to periodically pause the video player. To support synchronized viewing, we will develop an algorithm to automatically detect transitions between application screens. One a transition is detected, a screenshot of the next screen will be automatically sent to the server to find the next screenshot in the same page or the same video. Then, the enhanced viewer will automatically advance the content to the next screenshot.

## VII. Contributing

To complete the knowledge life cycle, the final research activity is to consider how we can create a more open platform to encourage more people to contribute online screenshot instructions. Systems for contributing knowledge to supplement existing material are common on the web (e.g. WebNotes and Shiftspace), where URLs and HTML page structure provide robust attachment points for users to link any article or video they generate to an arbitrary web page. But similar systems for the desktop have previously required application support. We plan to explore the use of screenshots as hooks to link applications to instructions. People who wish to contribute instructions to an arbitrary application can simply take a screenshot of the application and type some brief instructional text. The screenshot-text pair can be uploaded to our system to be indexed and made available to anyone. On the users’ side, we will build a tool that listens for hotkeys such as F1. Whenever the hotkey is pressed, the tool will automatically capture the screenshot of the most salient application screen on the desktop and lookup the instructions relevant to the application screen. This approach is open and accessible to anyone, since it does not require privileged access to the source code and programming experiences.

# Deliverables

At the conclusion of the project, we will contribute the following deliverables:

* A searchable repository consisting of at least one million screenshot instructions.
* An algorithm for detecting screenshot instructions.
* An algorithm for ranking screenshot instructions with respect to a query
* A lightweight client program for formulating and submitting multimodal queries.
* A plug-in for web browsers and video players to provide synchronized browsing for screenshot instructions.

# Evaluation

Each deliverable of the proposed project will undergo rigorous empirical evaluation.

* **Repository** What is the percentage of the pages or videos in the repository that are screenshot instructions?
* **Screenshot detection algorithm** What is the detection accuracy of our screenshot detection algorithm?
* **Ranking algorithm** We will use standard metrics such as precision-and-recall to evaluate the efficacy of the proposed ranking algorithm. We will compare to baseline approaches including but not limited to keyword-only ranking, image-only ranking, and no ranking. We will use the same evaluation methods used by the RankSVM paper.
* **Querying tool** How many words can be saved as a result of using screenshots as queries? How much faster can users specify queries? How much more successful can users formulate the right multimodal queries to get the results they want, compared to keyword only baselines? How easy is it to learn to specify multimodal queries? Do users improve their performance metrics over the course of a trial?
* **Reading tool** How fast can users follow screenshot instructions to perform tasks with and without our tool?

# Appropriateness for the III Program

The proposed research project is appropriate for the Information Integration and Informatics Program for the following reasons:

* This project **focuses on the processes and technologies** involved in creating, managing, visualizing, and understanding a useful class of digital content---online screenshot instructions.
* This project **addresses** **all stages of the knowledge life cycle** including creating, aggregating, indexing, querying, ranking, previewing, and reading.
* This project **involves multi-disciplinary collaboration** among qualified researchers of information technology, computer vision, and HCI where data (i.e., visual computer instructions) plays a central part.
* This project **is grounded in a useful application domain** that can benefit a diverse group of users in learning various computer-related tasks.

# Intellectual Merits

* We will understand how people use multimodal queries to search multimodal documents when both modalities are essential.
* What are the scientific questions that will be answered?
* TODO

# Broader Impacts

* Will impact a broad range of users on the way they access and learn computer knowledge from online screenshot instructions. While the proposed project doe not target a particular demographic group, we do expect some demographic groups such as young children and older adults will benefit even more from the easier access to screenshot instructions as promised by the project.
* Will pave the way to study other kinds of graphically illustrated instructions, such as instructions on cooking, car repair, and sports.

# Preliminary Studies

We did two preliminary studies to evaluate the potential of the proposed project in terms of technological feasibility and usability.

For technological feasibility, we studied the idea using a small collection of screenshot instructions contained in electronic books. We were able to apply standard image-matching techniques to index more than 50,000 pages. A significant number of these pages do contain screenshots and can be considered as examples of screenshot instructions. We found multimodal indexing using regular words, visual words, and OCR words provided the best retrieval performance based on a small test set of 500 queries. We shared the result as a poster at SIGIR 2009.

For usability, we recruited 15 subjects to try a Web-based simulator to experience taking screenshot and typing keywords to search for screenshot instructions. All the subjects we studied found the multimodal search method novel and easy to learn. We also measured statically significant reduction in task completion time in formulating multimodal queries than in formulating keyword-only queries. These findings point to the potential usability of a search system fully taking advantage of the multimodal nature of screenshot instructions. We have published the result as a part of a full paper publication at UIST 2009.

Encouraged by these preliminary findings, we seek funding from the III program to expand the project to the Web-scale to have broader impacts on the way we manage and access screenshot instructions.

# Work Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Fall 10** | **Spring 11** | **Fall 11** | **Spring 12** | **Fall 12** | **Spring 13** |
| **Aggregate** | R&D | Evaluate |  |  | Integrate & Deploy | Evaluate & Publish |
| **Index** | R&D | Evaluate |  |  |
| **Query** |  | R&D | Evaluate |  |
| **Rank** |  | R&D | Evaluate |  |
| **Preview** |  |  | R&D | Evaluate |
| **Read** |  |  | R&D | Evaluate |
| **Contribute** |  |  | R&D | Evaluate |