A Streaming Digital Ink Framework for Multi-Party Collaboration

Rui Hu, Vadim Mazalov, and Stephen M. Watt

The University of Western Ontario
London Ontario, Canada N6A 5B7
{rhu8,vmazalov,Stephen.Watt}@uwo.ca

Abstract. We present a framework for pen-based, multi-user, online collaboration in mathematical domains. This environment provides participants, who may be in the same room or across the planet, with a shared whiteboard and voice channel. The digital ink stream is transmitted as InkML, allowing special recognizers for different content types, such as mathematics and diagrams. Sessions may be recorded and stored for later playback, analysis or annotation. The framework is currently structured to use the popular Skype and Google Talk services for the communications channel, but other transport mechanisms could be used. The goal of the work is to support computer-enhanced distance collaboration, where domain-specific recognizers handle different kinds of digital ink input and editing. The first of these recognizers is for mathematics, which allows converting math input into machine-understandable format. This supports multi-party collaboration, with sessions recorded in rich formats that allow semantic analysis and manipulation of the content.

Keywords: Pen computing, distance collaboration, mathematical handwriting recognition, InkML, Skype, Google Talk

1 Introduction

We are interested in development of an infrastructure that helps researchers, engineers, teachers and students to collaborate online over pen-based and graphical interfaces. Pen-based collaboration in mathematical domains can significantly increase productivity, e.g. Gowers, et al. conducted an experiment of "attacking" a mathematical problem through a collaboration of volunteers online [1]. In just over five weeks, more than two dozen individuals contributed approximately 800 comments that led to the solution of the problem. We believe that the synergy of pen-based *collaboration* and *recognition* of mathematical input can enhance the efficiency of online interaction. Nevertheless, there is no technology that allows to capitalize on both simultaneously: some software handles recognition without the ability for real-time sharing, e.g. the Maple computer algebra system [2], while other systems provide a whiteboard for collaboration, but no mathematical recognition, e.g. Microsoft OneNote [3], Calliflower [4] or Dabbleboard [5].

We present a framework for multi-user online collaboration in a pen-based and graphical environment. This environment allows participants conducting and archiving collaborative sessions that involve synchronized voice and digital ink on a shared canvas. The digital ink is represented as InkML [6], allowing special recognizers for different content types, such as mathematics and diagrams. The collaborative sessions may be recorded and stored for later playback, analysis or annotation. The framework currently employs the popular Skype [7] or Google Talk [8] services as the backbone to deliver data streams, but other transport mechanisms could be used. Our objective is to support computer-enhanced distance collaboration, where domain-specific recognizers handle different kinds of digital ink input and editing. The first of these domains is mathematical input, which has similarities to both natural language handwriting input and two-dimensional diagramming. This framework has potential to increase productivity in teleconferences or to enhance online learning and tutoring, as the participants can interact in a natural way. A version of the framework implementation is available for download at `http://www.orcca.on.ca/InkChat/`. The current release (version 0.9.5) has many of the features described in the paper. Some of the features described here have been implemented in separate packages and will be integrated in later versions of InkChat. This work is an outgrowth of earlier work [9] that allowed collaborative inking, but which did not allow modular extension.

The remainder of this article is organized as follows. In Section 2, we explain why it is important for collaboration software to be portable. We then investigate potential challenges that may be encountered in the development and propose our solutions. Section 3 presents a high-level overview of the architecture of the framework and gives details on each component. In Section 4, we describe the implementation of the framework. A case study and a discussion of lessons learnt are given in Section 5. Section 6 concludes the article.

2 Portability of the Framework

A number of digital ink applications have been developed over the past years, following the emergence of digital ink. Most of these applications lack support for portability and are consequently each restricted to a single platform. Collaboration software, however, is most useful when it is platform-independent, both because an individual may use different platforms at different times, and because teams may be composed of members using different systems. Dealing with significantly different client software reduces the ability of the individual or the group to master its use. If any of the members has difficulty, efficiency of the whole team is reduced. Those tools that are cross-platform usually can work only with proprietary ink standards and thus are restricted in their ability to share data with other applications. In this section we investigate the challenges of building a portable framework.

2.1 Portability of Software

Today's platforms capture digital ink in various data structures and process it using different mechanisms. Our framework must therefore deal with different

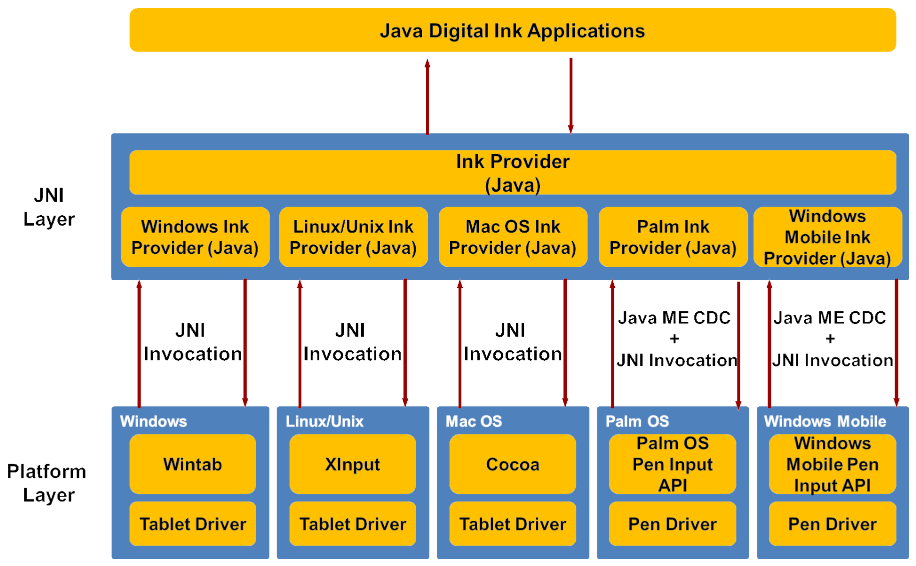


Fig. 1. A cross-platform framework for digital ink applications

lower-level interfaces on different platforms. On Windows, digital ink is captured in Wintab packets or `Stroke` objects and transmitted by the Wintab interface or the Windows XP Tablet PC APIs. A similar scheme is applicable to Linux platforms that use `XInput` events and the Linux Input Subsystem. For Mac OS X we use the `NSEvent` objects and the Cocoa Framework. These platform-specific APIs make development of pen-based collaboration software difficult.

Having explored available APIs on a variety of platforms, including Windows, Windows Mobile, Linux, Mac OS X and Palm OS, we use a framework that can capture digital ink across these platforms and provide a platform-independent, consistent interface to digital ink applications. This framework, as illustrated in Figure 1, contains two layers: the platform layer and the Java Native Interface (JNI) layer. The platform layer receives digital ink input from drivers and passes the data to the upper interfaces: Wintab for Windows, XInput for Linux/Unix and Cocoa for Mac OS. These interfaces push the data to the user space in a platform-specific way and describe each data event inconsistently. This puts the responsibility for event conformance on the shoulders of developers of ink applications. One of our objectives is to allow developers to focus on functionality, rather than the platform-specific issues. This leads to the design of the JNI layer.

The JNI layer interacts with the platform layer and provides consistent, platform-independent APIs for digital ink applications. As the input APIs are implemented in C-like languages, the JNI layer can invoke them using the Java Native Interface. The JNI layer collects digital ink input from the platform layer, converts it to platform-independent events and then dispatches to digital ink applications. This allows development of pen-based applications on top of the JNI layer to be platform-independent.

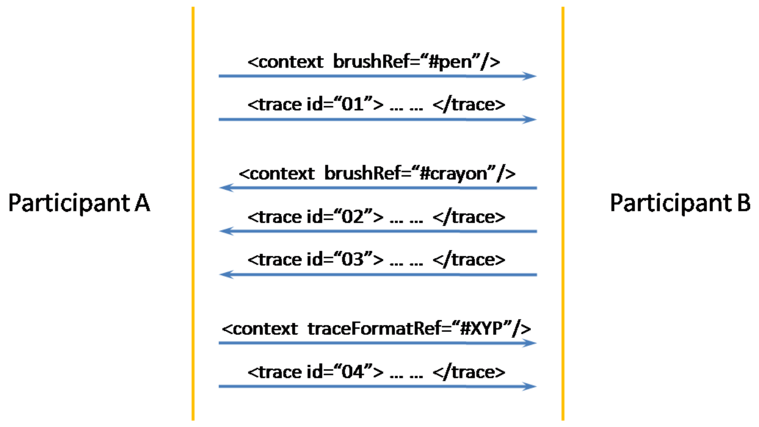

Participant A
Participant B
<context brushRef="#pen"/>
<trace id="01"> … … </trace>
<context brushRef="#crayon"/>
<trace id="02"> … … </trace>
<trace id="03"> … … </trace>
<context traceFormatRef="#XYP"/>
<trace id="04"> … … </trace>

Fig. 2. Ink streaming

2.2 Portability of Digital Ink Data

Another challenge in developing whiteboard-diagramming software is associated with the heterogeneous environment. Various pen devices have different characteristics, including sampling rate, channel properties, screen resolution and so on. The representation of the data generated by a device affects the usability of an application. Previous whiteboard tools did not address this as thoroughly as we require. An example is InkBoard [10], a collaborative sketching application based on Microsoft's Conference XP research platform [11] and designed for Tablet PCs. InkBoard allows design teams to interact by streaming digital ink in Microsoft's proprietary Ink Serialized Format (ISF) [12]. As a result, the application is limited in use to Windows environments where ISF is natively supported.

To represent digital ink, we find it useful to adopt an open, flexible, powerful, platform-, and vendor-independent standard, such as InkML [6]. This allows complete and accurate representation of digital ink by capturing the recording information such as the device characteristics, pen tilt, pen pressure and so on. Most importantly, it provides support for collaboration applications that require streaming ink between participants.

The InkML streaming of digital ink is based on the concept of "context." Whenever digital ink is written, there is some context in effect. Contexts may be represented externally using the `<context>` element in InkML. This can contain various associated attributes, including canvas properties, canvas transformation, trace format, ink source metadata, and time stamp. Initially, each ink collaboration participant obtains a default context and listens for context changes. This is similar to an event-driven model in which context changes are made when contextual elements are received. In practice, these elements will intersperse among digital ink streams, as shown in Figure 2. With this model, each participant can easily maintain the current context of the sender in addition to

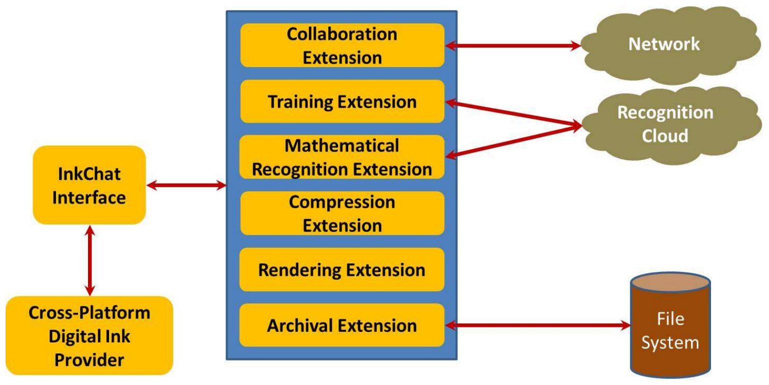


Fig. 3. InkChat architecture

its own local context. Whenever a new contextual element is received, it simply updates old values. Contextual elements are sent only when there is a context change, which helps to decrease streaming overhead on the wire. The overhead can be further reduced by making references to existing contextual elements, which can be pre-defined or previously received. This can be accomplished by using referencing attributes (e.g. `brushRef`) of `<context>`.

3 Architecture

We now present the framework for multi-user online collaboration, allowing sessions to be recorded in formats that allow semantic analysis and manipulation of the content. The framework currently consists of InkChat, a digital ink application developed at Ontario Research Centre for Computer Algebra, and a number of extensions. InkChat is the main platform of the application, on top of which other extensions may be added. Figure 3 presents a high-level overview of the architecture of InkChat. InkChat interacts with the cross-platform framework, presented in Section 2.1, whose primary purpose is to collect digital ink from a variety of platforms and to provide a platform-independent, consistent interface for digital ink applications. The six extensions, most of which can work independently and simultaneously, serve as plug-ins for InkChat. We outline the details of each extension below.

3.1 The Collaboration Extension

Similar to the traditional concept of sharing a session between several parties available in communication software (e.g. having a group call or a text chat), the collaboration extension allows real-time sharing of the canvas among participants and enables the participants to edit content on the canvas. Synchronization of the canvas is performed through the underlying communication backbone. In

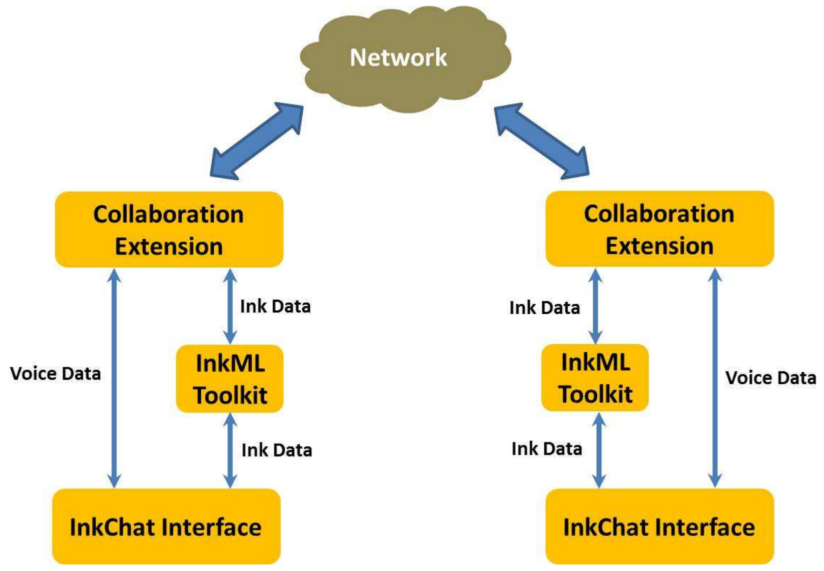


Fig. 4. Collaboration framework

addition, the voice and video channels are typically available through the communication software as well. The collaboration scheme between two clients is shown in Figure 4. Digital ink data is first converted into InkML format and then sent in a data channel parallel to the voice channel. The collaboration extension itself is an abstract layer that can handle different network protocols including pipes and sockets in P2P (peer-to-peer) and the traditional client-server configurations. To adapt to the collaboration environment, we allow participants to choose the most suitable protocol at the beginning of the conversation. For example, if the collaboration is intended to take place in a small group and requires only the basic functions of whiteboard, P2P would be preferable as it is inexpensive and fairly simple to set up and manage. If the collaboration is intended to comprise a large number of participants and demand sophisticated computing services (e.g. mathematical formula simplification), client-server mode may be more appropriate.

InkChat supports conference mode where more than two participants can be involved in one conversation. Depending on the chosen network protocol, InkChat employs different mechanism to communicate with other participants. When a P2P backbone such as Skype is used, the conference is initiated by the host that has a connection with every other participant. Digital ink routing shares the same mechanism as audio routing, each ink stroke will be broadcast by the host to all participants except the initiator. Figure 5 shows the two cases, when a host and when a client initiate a stroke. In the client-server network, the server will play the role of the host and establish a connection with each client, as shown in Figure 6. Both the digital ink and audio data are sent to the server first and then broadcast to all the other clients.

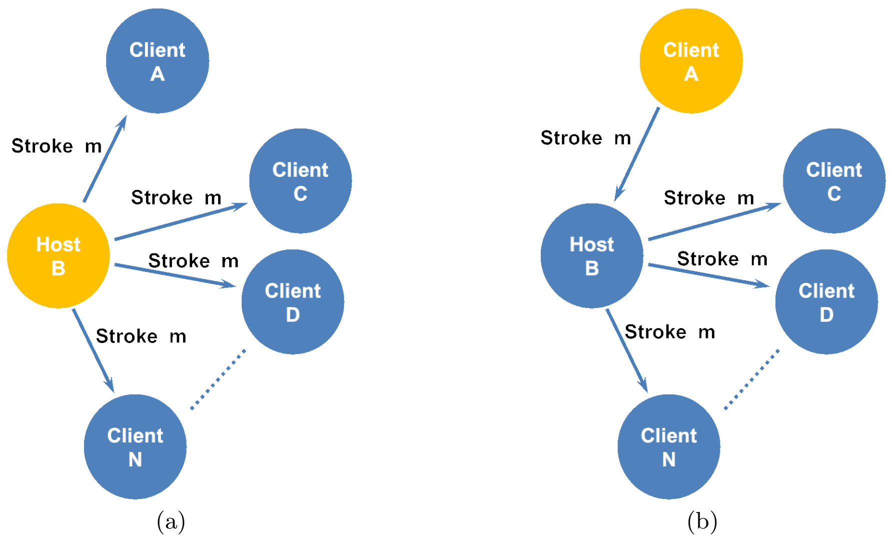


Fig. 5. Sessions with the stroke initiated by (a) the host and (b) the client.

3.2 The Training Extension

In an adaptive recognition environment, a separate training phase is not required. Having some number of training samples in each class can, however, significantly improve the initial recognition. The number of training samples that a class should contain depends on the recognition methods. Using our approach, the recognition rate depends on the minimal distance to convex hulls of neighbouring classes. For most of the classes in our dataset, about 20 classes are required [13].

A training session may be initiated on first use of the application or when a new character is first introduced. Once training is finished, a profile of classes and training samples is saved as an XML document. The profile is a hierarchical container of catalogs (groups of related characters, e.g. digits, Latin letters), symbols, writing styles, and training samples [13]. Training samples are divided by different forms for symbols (e.g. a one-stroke versus a two-stroke numeral "ϕ"), since many recognition methods are sensitive to the direction and order of strokes. The training extension also provides an interface for the user to manage profiles of training samples.

3.3 The Mathematical Recognition Extension

The mathematical recognition extension is an ongoing project. The objective is to have domain-specific recognizers that handle different kinds of digital ink input. We currently focus on the classification of handwritten mathematical symbols, as the essential component of math formula recognition, and spatial analysis of characters. In our classification paradigm, each character is represented as a

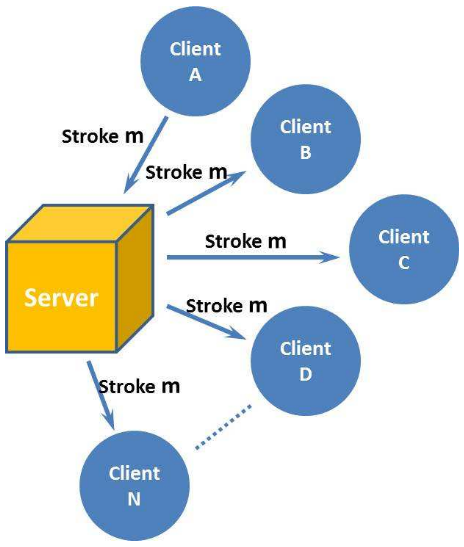


Fig. 6. A client-server configuration with a stroke initiated by a client.

single point in a space of curves. The coordinates of this point are the coefficients of truncated orthogonal series approximating the coordinate functions of the trace [14]. Classification of a character is based on the distances to the convex hulls of nearest neighbours in this space. A sequence of incoming strokes is divided to form characters with the highest classification confidence. Spatial analysis of samples is performed based on the relative positioning of the centers of mass of adjacent characters.

This approach typically does not require many training samples to discriminate a class. However, because there is a large number of classes, the training dataset may contain tens of thousands of characters. The dataset evolves with each recognized sample. Synchronization of the evolving per-user training exemplars across several pen-based devices may become tiresome. To address this aspect, the storage of the training database can be delegated to a cloud. [13]. At present, recognition is always done locally, although this could alternatively be done by a server. Locally, the recognition extension accepts raw ink from InkChat and preprocesses it. This preprocessing typically includes computing approximation of the character and normalization with respect to the size and position. Consecutive strokes are merged into candidate symbols and approximated, yielding points in the space of curves. The coefficients are recognized [14] by the recognition extension. A ranked list of recognition results is sent to InkChat for display and confirmation or rejection/selection. The result is stored remotely as classification experience for subsequent adjustment of training clusters.

3.4 The Compression Extension

The compression extension implements a hybrid of functional [15] and linear [16] approximation. There are several schemes for segmentation of digital ink for compression by the functional approximation method. The most robust is the adaptive segmentation that dynamically selects the degree of approximation and the size of coefficients. Coefficients are recorded as floating-point numbers with base 2 [15]. The functional approximation is best-suited for curly handwriting, for example as with math symbols.

Linear approximation allows compact representation of nearly linear segments, and is therefore suitable for many forms of representation of mathematical and engineering knowledge, such as graphs, tables, or diagrams. The method removes points that least affect the shape of the curve, so long as the error between the original and the approximating curves remains within a threshold. The method can be viewed as a dynamic adjustment of the density of points, depending on the shape of the stroke. The method is very fast and yields reasonable compression.

3.5 The Rendering Extension

Different rendering styles are required for different drawing and writing activities. For example digital painting and Chinese calligraphy may require an instrument that behaves like a paint brush, while diagramming may best be done with an instrument that behaves like a pen or pencil. To support these needs, different brush models may be selected.

Round Brush The round brush, as its name suggests, draws each ink point as a filled circle. We use three parameters to model the round brush: x, y to indicate the position and r to measure the circle radius which can be a function of the pen tip pressure. We fill the gaps between the circles of consecutive points by forming envelopes with circle tangents, as shown in Figure 7(a).

Tear Drop Brush The idea of the tear drop brush is to model the contact of a brush head as it varies in distance to the canvas and is dragged along. This contact area is modelled as tear drop – a circle and the area enclosed by the tangents to a trailing point (which may degenerately be on the circumference). Varying the radius models what happens when the brush varies in distance to the canvas, and the trailing point is determined by the past history of the brush. There are five parameters, as shown in Figure 7(b): x and y give the position of the ink point, r gives the head radius, θ the direction of the tail, and ℓ the length of the tail from (x, y). The tear drop brush model has been adopted by InkML [6] and the calculation of r, θ and ℓ from the stream of x, y and pressure values has been discussed in [17, 18]. As with the round brush, strokes are rendered by filling brush shapes and the area formed by tangents between successive brush outlines.

3.6 The Archival Extension

InkChat stores handwritten input using InkML format. Strokes are converted to an InkML stream as they are captured. They are then saved to the current

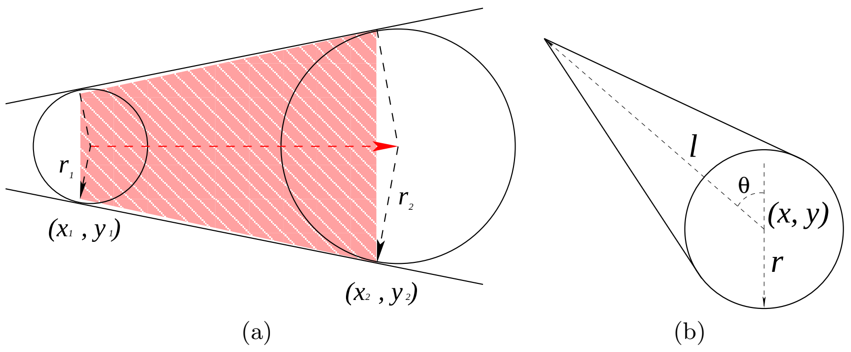


Fig. 7. The brush models: (a) the round brush and (b) the tear drop brush.

ink session before being sent to other participants. As the stream is received by a participant, InkChat immediately parses the stream and saves the strokes to the current ink session. When the conversation is finished or the user requests to save, InkChat writes the current ink session to an InkML file with the ink from all participants. InkChat also supports loading collaboration sessions from InkML documents.

4 Implementation

We have used the framework presented in section 3 to build the InkChat application. Most components are implemented in Java as it provides strong portability and applications can run on any Java Virtual Machine regardless of system architecture. For platforms that do not support Java directly, the Java code can be compiled to C. Our goal to maintain a single, coherent source that can be compiled for all platforms. Below we briefly describe the implementation of the major components.

4.1 User Interface

The InkChat user interface is illustrated in Figure 8. It is designed to have buttons grouped so that the distance of moving the pen is minimized. Users can write, erase, and highlight by selecting corresponding brushes. Editing is also supported. Users can redo, undo, cut, copy, and paste different kinds of content, such as images, typed-text, and digital ink. In order to better support collaboration, we have provided a floating pointer and a page navigator. The floating pointer can be used to point at target objects on the shared canvas without leaving ink. Together with voice channel, this allows pointing to and discussing aspects of the common canvas. The page navigator allows participants to create new pages or review earlier pages. Once a session is finished, all pages can be recorded and stored for later playback.

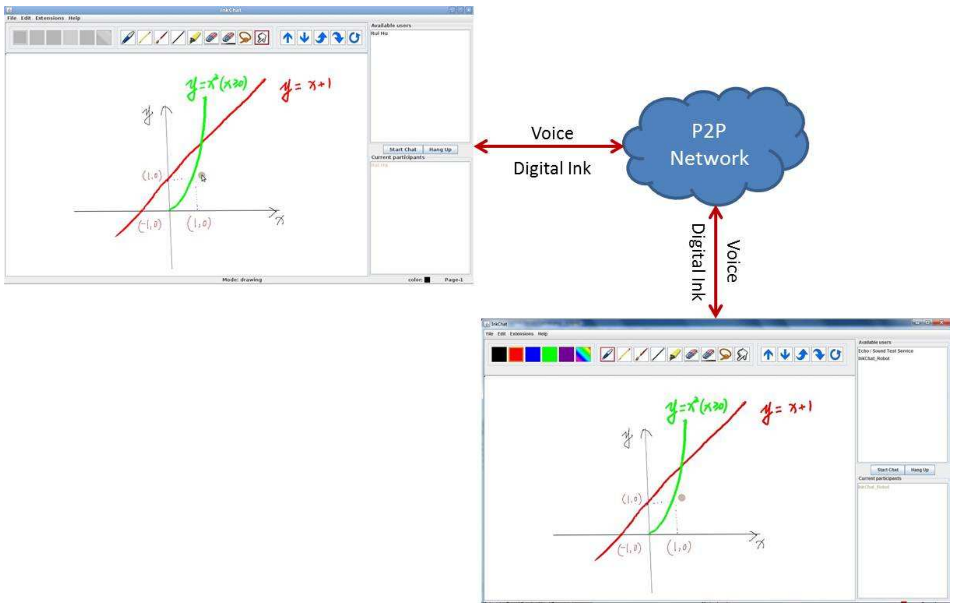


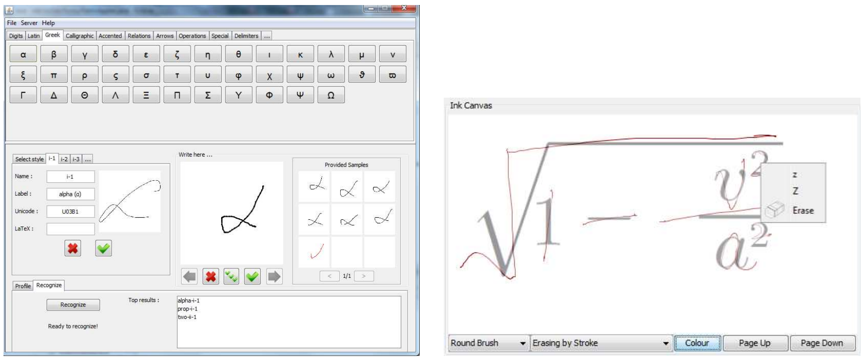
Fig. 8. An example of diagramming in InkChat with Skype service.

4.2 Collaboration

The InkChat software combines digital ink and voice in a multi-user conversation with a shared canvas. It is structured so it can use the popular Skype and Google Talk services for transmitting data streams. InkChat can be used by people working together in the same room or on opposite sides of the planet. Sessions may be archived to be resumed later or for later processing. An example of collaboration using InkChat is illustrated in Figure 8. Two participants, the author and user `InkChat_Robot`, are involved in the collaboration using Skype. The author is working on a Windows machine while the `InkChat_Robot` is on a Linux machine. The top left screenshot shows the `InkChat_Robot`'s canvas and the bottom right screenshot shows the author's canvas. Remarkably, the `InkChat_Robot` is using the floating pointer to point around coordinate $(1, 1)$. This is streamed to the author's canvas in real time.

4.3 Training

The training extension presents an extensible catalog of symbols and styles, organized in a tabbed panel, as shown in Figure 9(a). Each tab contains a list of symbols. Once the user selects a symbol, the panel with styles becomes available. Styles are shown as animated images for visualization of stroke order and direction. Each sample is associated to an existing or new style. If a style has not been selected, it is determined automatically based on its shape and the number of strokes. All the elements of the interface (catalogs, symbols, styles



(a) (b)

Fig. 9. The recognition interface: (a) the training extension and (b) the mathematics recognition extension.

and samples) are dynamic: A context menu is allows the user to create, delete or merge elements. A profile can be saved locally or synchronized with a server.

4.4 Mathematical Recognition

Classification of symbols takes place when a user performs handwritten input through InkChat and the recognition extension is enabled. For each character, a context menu is available that lists the top recognition candidates, see Figure 9(b). If the user chooses another class from among the candidates listed in the context menu, adjacent characters can be reclassified based on the new context information. There are two usual approaches to expression recognition: character-at-a-time (each character is recognized as it is written) and expression-at-a-time (characters are recognized in a sequence, taking advantage of the context) Our current approach lies between these: we are experimenting with recognition within a moving window of context and consider all ink outside that window to be "dry" and recognized. Classification results can be displayed super-imposed on the digital ink or can replace it.

5 Discussion

5.1 Scenarios

Here we describe several cases in which the framework has been found useful.

Online learning and tutoring Figure 10 shows the triangle inequality being discussed by three persons. The top-right panel lists the available friends of the user and the lower-right panel shows the participants who are currently in the session (excluding the user). The Google Talk service is employed in this session. The triangle was drawn by the participant `Robot1 InkChat`. It was later annotated by the user to better explain the triangle inequality.

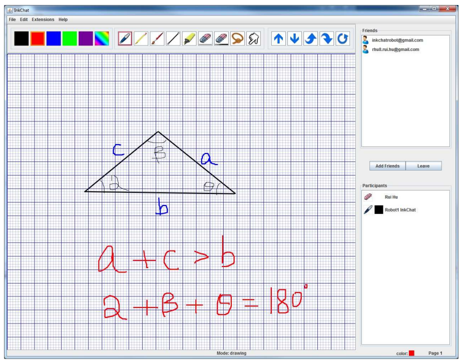


Fig. 10. An example of online learning and tutoring

Collaborative work on math and diagrams The framework provides a collaborative environment where participants can interact through a shared canvas and a voice channel. Figure 11 shows examples in different domains. The ink and meta-information is shared between participants through Skype or Google Talk.

Demonstration of animated diagrams At least in some cases, animation can significantly improve understanding of certain complex diagrams [19]. The InkChat framework allows digital ink to be animated, reproducing the original writing sequence, or sequenced in some other manner. Thus, animated sketched diagrams can be created and shared in real-time or made available for download.

5.2 Lessons Learnt

The architecture we have presented is a multi-year ongoing project, and there are several important lessons that we have learnt

- **Lesson 1:** *Anticipate that the world will keep changing.* It is important to have the data streams for an application well specified so new technologies, such as JavaScript and HTML 5 canvas can participate.
- **Lesson 2:** *There is power in less capable, but more uniform, interfaces.* We initially had different interfaces for ink sharing, recognition and the other behaviours. Having a common interface, while not perfectly adapted to any of these tasks, allowed them to be combined flexibly in useful and unanticipated ways (e.g., combining the recording/playback and recognition modules).
- **Lesson 3:** *Plan on sharing.* The extensions and the main platform should be designed to allow sharing of the functionality with third parties through the mechanism of web services. For example, the recognition extension should be able to accept a SOAP message with coefficients of a character to be classified and send back the recognition candidates.
- **Lesson 4:** *Don't pin the user down.* Plan on individuals making use of many devices. This implies a protocol for network storage of user-specific information, such as personalized handwriting recognition data.

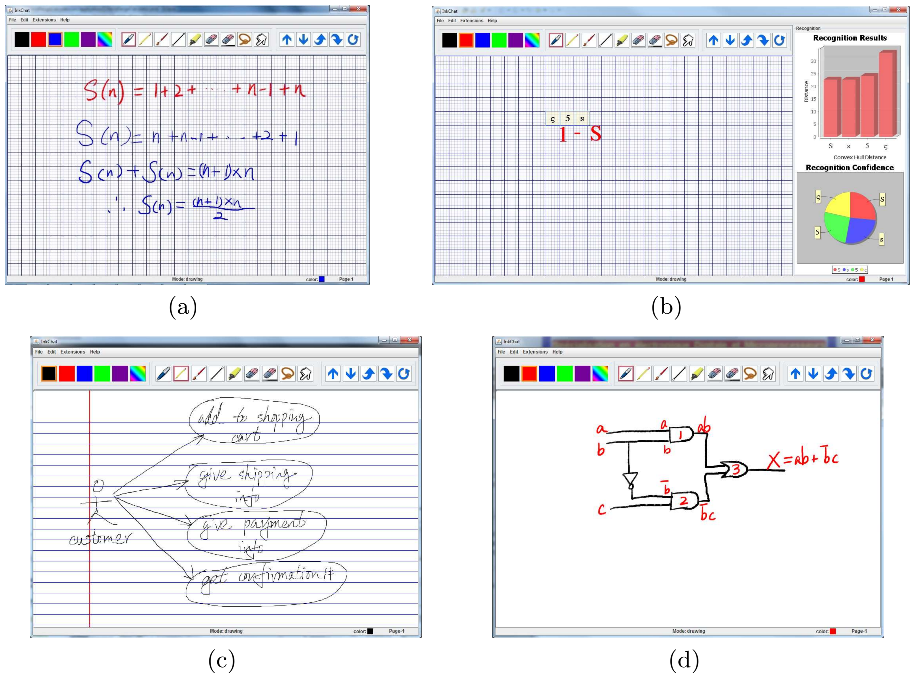


Fig. 11. Examples of math and diagram collaboration (a) formula induction, (b) math symbol recognition, (c) use case diagram, (d) circuit diagram.

- **Lesson 5:** *Anticipate standards.* Tracking draft standards prior to their maturity, and indeed participating in the standardization process, can be an effective strategy to achieve interoperability. The project was using InkML before it was recommended as a standard by W3C. Today InkML allows cut-and-paste of digital ink between applications, e.g. between Microsoft Office 2010 and InkChat.

6 Conclusion and Future Work

We have presented a framework for pen-based, multi-user, online collaboration applicable to mathematical domains. The framework supports teamwork on scientific and engineering problems by allowing participants to make contributions to a shared canvas while having a discussion. The framework is portable and uses InkML, a W3C standard, as a representation format. The architecture of the framework is extension-based with InkChat as the main component. On top of InkChat, several extensions have been developed for collaboration, training, mathematical recognition, compression, rendering and archival. We have presented a high-level overview of these components. In ongoing work, we are particularly interested in improved recognition of mathematical formula and architecting more powerful combinations of extensions.

References

1. Gowers, T., Nielsen, M.: Massively Collaborative Mathematics. Nature **461** (2009) 879–881
2. Maplesoft Inc.: Maple user manual
3. Microsoft Inc.: Onenote 2010
4. Iotum Inc.: Calliflower. `http://www.calliflower.com/`
5. Dabbleboard Inc.: Dabbleboard. `http://www.dabbleboard.com`
6. Watt, S., Underhill, T.: Ink Markup Language (InkML). `http://www.w3.org/TR/InkML/`
7. Microsoft Inc.: Skype. `http://www.skype.com/`
8. Google Inc.: Google Talk. `http://www.google.com/talk/`
9. Regmi, A., Watt, S.M.: A Collaborative Interface for Multimodal Ink and Audio Documents. In: Proc. 10th International Conference on Document Analysis and Recognition. ICDAR 2009, IEEE Computer Society (2009) 901–905
10. Ning, H., Williams, J.R., Slocum, A.H., Sanchez, A.: InkBoard - Tablet PC Enabled Design oriented Learning. In: Proc. 7th IASTED International Conference on Computers and Advanced Technology in Education. CATE 2004, ACTA Press (August 16-18, 2004) 154–160
11. Beavers, J., Chou, T., Hinrichs, R., Moffatt, C., Pahud, M., Powers, L., Eaton, J.V.: The Learning Experience Project: Enabling Collaborative Learning with ConferenceXP. Technical Report MSR-TR-2004-42, Microsoft Research (2004)
12. Microsoft Inc.: Ink Serialized Format Specification. `http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/InkSerializedFormat(ISF)Specification.pdf`
13. Mazalov, V., Watt, S.M.: Writing on Clouds. In: Proc. 2012 Conferences on Intelligent Computer Mathematics. CICM 2012, Springer Verlang (July 9-14 2012)
14. Golubitsky, O., Watt, S.M.: Distance-Based Classification of Handwritten Symbols. International J. on Document Analysis and Recognition **13**(2) (2010) 113–146
15. Mazalov, V., Watt, S.M.: Digital Ink Compression via Functional Approximation. In: Proc. 12th International Conference on Frontiers in Handwriting Recognition. ICFHR 2010, IEEE Computer Society (November 16-18, 2010) 688–694
16. Mazalov, V., Watt, S.M.: Linear Compression of Digital Ink via Selection of Subsets of Points. In: Proc. 10th IAPR International Workshop on Document Analysis Systems. DAS 2012, IEEE Computer Society (March 27-29, 2012) 429–434
17. Hu, R.: Portable Implementation of Digital Ink: Collaboration and Calligraphy. Master's thesis, The University of Western Ontario, Canada (2009)
18. Watt, S.M.: On the Mathematics of Calligraphy (invited talk). International Conference On Mathematics Mechanization – In honor of professor Wen-Tsun Wu's ninetieth birthday, Beijing, China (2009).
19. Burd, E., Overy, D., Wheetman, A.: Evaluating Using Animation to Improve Understanding of Sequence Diagrams. In: Proc. 10th International Workshop on Program Comprehension. IWPC 2002, IEEE Computer Society (2002) 107