

修士論文 2019 年度（令和元年度）

手書きベース Wiki システムの研究

慶應義塾大学大学院 政策・メディア研究科

早川 匠

2020 年 1 月

修士論文 2019 年度（平成元年度）

手書きベース Wiki システムの研究

論文要旨

手書きメモやイラスト等のグラフィカルなデータに自在にハイパーリンクを埋め込み、それらを Wiki として利用できる「手書きベース Wiki」システムを提案する。手書きメモやイラストは広く浸透した情報の記録・表現手法であるものの、紙を前提としたフォーマットであるために参照や管理、再利用が難しいという問題が存在する。計算機上で手書きメモの作成や管理を行うツールは広く利用されているものの、これらは紙のメモやイラストの利用形態を再現したに過ぎず、この問題を本質的に解決していない。手書きベース Wiki は、ハイパーテキスト・ハイパーリンクや Wiki 等の技術の組み合わせによって、手書きでメモやイラストを描きながら、自在にハイパーリンクを埋め込んだり、ハイパーリンクによって関連する他のメモやイラストを簡単に参照できるシステムである。既存の手書きメモ・イラストの問題点を解決するだけでなく新しい活用法を提案するため、手書きベース Wiki のプロトタイプ「DrawWiki」を実装した。本論文では手書きベース Wiki としての DrawWiki の設計や評価、応用例について述べ、研究の発展性について考察する。

キーワード

手書きメモ、イラスト、Wiki、ハイパーテキスト、ユーザーインターフェース

慶應義塾大学大学院 政策・メディア研究科

早川 匠

Abstract Of Master's Thesis Academic Year 2019

A study on drawing-based wiki systems

Summary

We propose a drawing-based note-taking style where users can use handwritten objects not only for showing shapes and texts, but for linking objects just like hyperlink texts are used for linking pages. Hypertexts are widely used on wiki systems like Wikipedia, where words and phrases are used for linking pages. Although wiki systems are useful for managing a large amount of text data, it is not possible to use non-text data for linking information. It would be more useful if handwritten drawings can also be used as hyperlinks on wiki pages just like textual phrases are used for linking pages. To prove the concept of drawing-based wiki systems, we have implemented a prototype system “DrawWiki”, where arbitrary handwritten drawings can be used as links to other pages and objects.

In this paper, we describe the design, implementation, evaluations and applications of DrawWiki, and discuss the future of wiki systems where the mixture of texts and drawings are used as hyperlinks.

Keywords

Handwritten-notes, Illustration, Wiki, Hypertext, User Interface

Graduate School of Media and Governance
Keio University

Takumi Hayakawa

目 次

第 1 章 序論	1
1.1 研究の動機	2
1.2 研究の目的	2
1.3 本論文の構成	3
第 2 章 研究背景	5
2.1 手書きのメモ・イラスト	6
2.2 タッチ・ペンインターフェースの普及	6
2.3 計算機上で手書きメモ・イラストの作成支援	6
2.4 手書きデータを表現する画像ファイルフォーマットの制約	7
2.5 手書きメモ・イラストを扱う既存のシステム	7
2.5.1 メモアプリケーション	7
2.5.2 イラスト投稿・共有システム	8
2.6 テキストの進化	10
2.6.1 Wiki システム	11
2.7 手書きメモ・イラストの問題点	12
2.8 まとめ	12
第 3 章 設計	13
3.1 要件	14
3.2 ハイパーイラスト	14
3.2.1 ハイパーイラストの定義	14
3.2.2 ハイパーイラストの仕様	15
3.3 手書きベース Wiki	16
3.3.1 手書きベース Wiki の定義	16
3.3.2 DrawWiki	16
3.3.3 機能と使い方	17
第 4 章 実装	21
4.1 アプリケーション構成	22
4.2 クライアントサイド	22
4.2.1 手書きデータの取得	22
4.2.2 仮想 DOM の利用	22

4.2.3	debounce された更新処理	23
4.2.4	リンク付要素の視覚的表現	24
4.3	サーバーサイド	24
4.3.1	アプリケーションサーバー	24
4.3.2	アセットサーバー	25
第 5 章	応用例	27
5.1	ハイパーイラストを活用したメモ	28
5.1.1	ソフトウェアのプロトタイピング	28
5.1.2	自作の整備メモ	28
5.2	ナレッジの共有と共同編集	29
5.3	手書きによる Web ページ・Wiki の作成	29
5.4	まとめ	30
第 6 章	関連研究	31
6.1	主要な研究領域	32
6.2	手書きメモ支援システム	32
6.2.1	Notepal	32
6.2.2	InkSeine	32
6.3	デザイン・プロトタイピングツール	34
6.3.1	DENIM	34
6.3.2	SILK	35
6.3.3	IntaractivePaper	35
6.4	手書きデータを利用した Active Reading やコラボレーション	36
6.4.1	XLibris	36
6.4.2	SketchComm	36
第 7 章	考察	39
7.1	評価	40
7.1.1	筆者の運用経験	40
7.1.2	意見・感想	41
7.1.3	問題点・要望	41
7.2	考察	42
7.2.1	設計指針の妥当性	42
7.2.2	解決すべき課題	42

7.2.3 手書きメモ・イラストの問題点の検証	42
第8章 結論	45
8.1 研究の成果	46
8.2 総括	46
謝辞	47
参考文献	48

図 目 次

2.1 iPad	6
2.2 Surface	6
2.3 iOS のメモ	8
2.4 evernote	9
2.5 googlekeep	9
2.6 現在閲覧している作品	9
2.7 閲覧中の作品に関連する作品群	9
2.8 ニコニ・コモンズによって可視化されるコンテンツ間の親子関係	10
2.9 Scrapbox の画面	11
2.10 関連ページリスト	11
3.1 ハイパーイラストの概念図	14
3.2 SVG 上で表現される手書きストローク	15
3.3 DrawWiki の初期画面	16
3.4 ハイパーイラストの作成画面	17
3.5 プリセットを用いて描かれた図	18
3.6 範囲選択ツール	18
3.7 リンク埋め込み機能の操作画面	18
3.8 関連イラストの表示機能	19
3.9 関連イラストのモーダル	19
3.10 エキスポート機能の操作画面	19
3.11 埋め込まれたハイパーイラスト	19
4.1 アプリケーションの構成	22
4.2 PointerEvent API の概要	23
4.3 仮想 DOM のモデル	23
5.1 DrawWik を用いて作成された Drawwiki のモックアップ	28
5.2 アクションによってポップアップするダイアログの再現	28
5.3 DrawWiki による自作整備メモ	29
5.4 関連情報を表示した画面	29
6.1 Notepal の画面	32
6.2 InkSeine の操作画面	33

6.3	DENIM の画面	34
6.4	操作を行うタブレット機器	34
6.5	SILK の画面	35
6.6	IntaractivePaper の画面	35
6.7	操作中の XLibris	36
6.8		36
6.9		36

表 目 次

第1章 序論

本章では本研究の動機と目的、および本論文の構成について述べる。

1.1 研究の動機

手書きでメモを取ったりイラストを描いたりすることは、情報を記録し、表現する手段として一般的であるが、その様式は紙や鉛筆等の筆記具が発明された頃からほとんど変わっておらず、一枚の紙の上で表現する事を前提としているため参照や管理、再利用が難しいという問題点が存在する。

また計算機が普及した現在では、手書きメモ・イラストを紙の上ではなくデータとして作成するソフトウェアも広く利用されているが、それらはPNG¹やJPEG²のような一枚の絵をピクセルの集合で単純に置き換えた形式で記録されることが一般的で、紙に描かれたものと比較して本質的に変化していない。この制約により計算機上で作成したメモやイラストであっても、すぐに参照できるようにするために階層化やタグ付け等の運用上の工夫が要求され、紙の上で手書きのメモを取っていた時と同じ不便さを引き継いでいる。

一方で手書きメモやイラストと同じく紙の上で記録されていたテキストは、計算機の登場により以下のように変化した。

- 他の文書への参照を実現するハイパーリンクと、それを内包した文書であるハイパーテキストが登場した
- Webによって様々なハイパーテキストに手軽にアクセスできるようになった
- コラボレーションツールであるWikiが複数人による共同編集を可能にし、知見の共有を実現した

これにより参照や管理・再利用が難しいという問題が解決された。かつては手書きメモ・イラストと同様の問題を抱えていたテキストは、計算機による新しい活用法が発明された事で広く普及するに至った。そのため手書きメモ・イラストも、計算機を活用する事で問題を解決し、進化する余地があると考えられる。

1.2 研究の目的

本研究では、手書きメモ・イラストを扱う既存のシステムが抱える参照や管理、再利用の難しさといった問題を解決し、またハイパーテキストやWiki等の技術を取り入れることで従来のシステムでは実現できなかった手書きメモ・イラストの新しい活用法を実現するシステム「手書きベースWiki」の構築を目的とする。

¹<http://www.libpng.org/pub/png/>

²<https://jpeg.org/jpeg/>

1.3 本論文の構成

本論文は以下の 8 章で構成される。

第 2 章では、本研究の背景をより詳細に分析し、既存システムの問題点を整理する。

第 3 章では、本論文で提案するシステムの基本構成と使い方について述べる。

第 4 章では、本論文で提案するシステムの詳細な実装について述べる。

第 5 章では、本論文で提案するシステムによって実現可能な応用例について述べる。

第 6 章では、関連する研究を紹介し、それらの特徴や本研究との関連を述べる。

第 7 章では、筆者による運用経験やユーザーからのフィードバックをまとめ、本論文で提案するシステムの有効性と問題点について述べる。

最後に、第 8 章で本論文のまとめと結論を述べる。

第2章 研究背景

本章では手書きメモ・イラストを扱う既存のシステムの現状と、その問題点を整理する。

2.1 手書きのメモ・イラスト

手書きによるメモやイラストは情報を記録・表現する手法として広く普及している。筆記具と紙さえあればすぐに記録でき、また美麗な作品を描くことを目的としなければ、特別な技量も要求されないためである。計算機の登場によりテキスト編集支援機能が充実したため、文章のみで完結する内容であれば手書きではなくテキストとして記録するように置き換わったが、アイデアのような文章のみでは表現しづらい構造を持った概念を表現する場合は文字と図を自在に混合させて配置できる手書きメモの方が適している。また、テキストによってメモをとる場合はキーボードのような専用のハードウェアや、それらを使いこなすタッチタイピング等の技量が必要であるという問題点があるが、手書きの場合は紙やペン等の筆記具が扱えれば良いため、ハードウェアや技能を必要とするテキスト入力と比較してより多くの人々が利用できる手段であると言える。

2.2 タッチ・ペンインターフェースの普及



図 2.1: iPad

図 2.2: Surface

かつてはノートやスケッチブック等の紙の上で記録されていた手書きメモだが、タッチパネルやスタイラスペン等のインターフェースを備えたデバイスの普及に伴い、計算機上で手書きメモを取ることが一般化してきた。手書きメモやイラストを計算機の上で描く場合、マウスやトラックパッド等のポインティングデバイスではなく、スタイラス等のペンインターフェースが好ましいとされるが、iOS¹やWindows²、ChromeOS³等の主要なプラットフォームでスタイラスペンを備えた機種が充実しているため手書きでメモやイラストを描く環境は充分に整っていると考えられる。

2.3 計算機上で手書きメモ・イラストの作成支援

計算機上でメモやイラストを作成する場合、以下のような編集支援機能を利用することができます。

¹<https://www.apple.com/jp/ios/>

²<https://www.microsoft.com/ja-jp/windows>

³<https://www.google.com/chromebook/>

- コピーやペースト
- Undo や Redo
- オブジェクトの移動や変形

これらの機能は紙という物理的なメディアの上では実現不可能であったが、現在では手書きデータを扱う大抵のアプリケーションが備えている。計算機の進化によって、手書きメモ・イラストの作成については編集支援機能によって利便性が大きく向上したと言える。

2.4 手書きデータを表現する画像ファイルフォーマットの制約

一方で作成した手書きメモを保存・記録する画像ファイルのフォーマットの機能は大きく制限されている。一般的に出力先として JPEG や PNG 等のビットマップ画像が用いられているが、この種のフォーマットは描いたもの全てがピクセルの集合として統合・変換されるため、インクをピクセルに置き換えただけのファイルとして保存される。したがって画像ファイルを参照するためには、

- 命名規則に則ってファイル名を決める
- 階層構造を設定してフォルダ分けする
- タグを複数設定してファイルにラベリングする

等の管理・運用上の工夫が必要となる。手書きメモを計算機で作成することは可能になったが、その参照や管理の不便さ、再利用の難しさは紙の上でメモをとっていたころからあまり改善されていない。

2.5 手書きメモ・イラストを扱う既存のシステム

2.5.1 メモアプリケーション

手書きデータを効率よくメモとして扱うこととするシステムを解説する。

(1) iOS/Mac のメモ

Apple の iPad や Mac にはメモアプリケーションが標準でインストールされており、指や Apple Pencil を用いて素早く手書きメモを取ることができる。一方で描いた手書きメモは画像として保存されるのみで、それを後から検索する機能は存在しないため、参照できるようにファイル名を決める、もしくは保存するフォルダを分類する等の工夫が求められる。

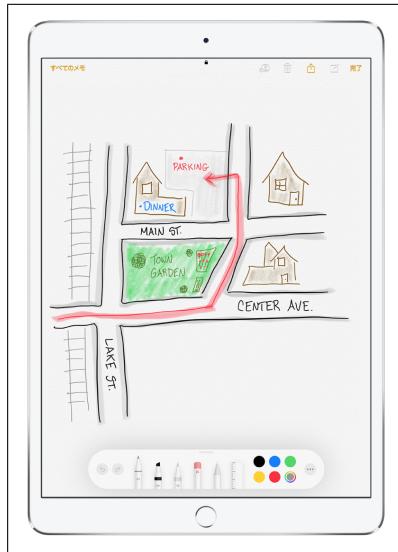


図 2.3: iOS のメモ

(2) Evernote

Evernote Corporation が開発する Evernote は指やスタイラスペンで手書きのメモやスケッチを作成できるほか、手書きメモ内の文字を認識し、全文検索によって手書きメモを参照する機能を備えている。あくまで検索対象は認識できた手書き文字のみであり、図形や描いたものの形状等のグラフィカルなデータから手書きメモを参照することはできない。そもそも Evernote において手書きメモを含めた全てのメモはノートブックと呼ばれる階層に分類して管理することを前提としているため、今取っているメモはどのノートブックに属するものかを予め意識する必要がある。

(3) Google Keep

Google が開発する Google Keep も、Evernote と同様に手書き文字を認識し、テキスト検索を行う機能を備えている。Evernote と異なり、Google Keep にはフォルダ等の階層構造はなく、全てのメモがフラットに管理される。必要に応じてメモに色やラベルをつけることによって、複数のメモを分類できるようデザインされている。

2.5.2 イラスト投稿・共有システム

Web 上に投稿・共有されたイラストを、効率よく参照できるよう管理することを目的としたシステムを解説する。これらのシステムは基本的に完成されたアート作品を投稿するのが主な用途であり、メモを素早く記録する手段として使われることはない。

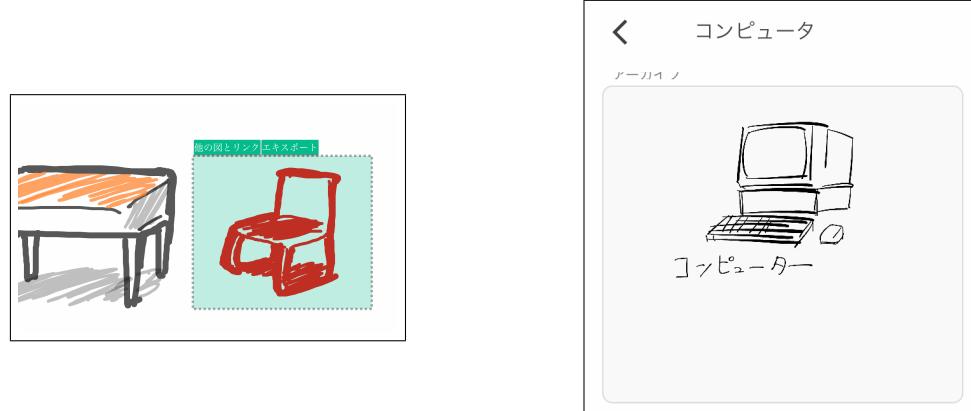


図 2.4: evernote

図 2.5: googlekeep

(1) Pixiv

pixiv Inc. が開発する Pixiv では、投稿したイラストに複数のタグを付加することができる。また共通のタグを持つ他のイラストを関連イラストとして下部に表示する機能を備えているため、作者を横断して共通するテーマの他のイラストを参照することができる。基本的にタグの編集は手動で行われるため、タグ付けするキーワードによっては表記ゆれが生じ、適切に検索できない場合もある。

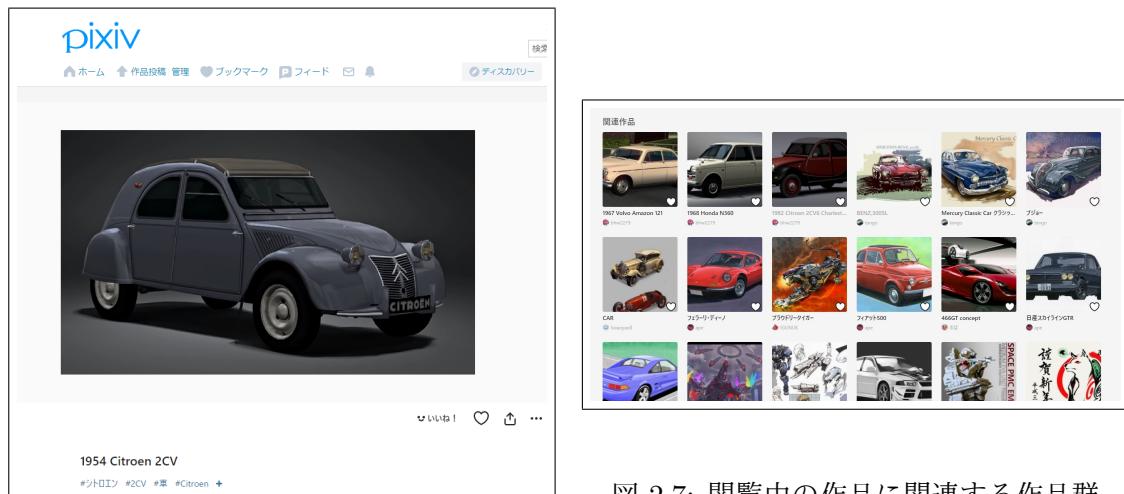


図 2.6: 現在閲覧している作品

図 2.7: 閲覧中の作品に関連する作品群

(2) ニコニ・コモンズ

ドワンゴが開発するニコニコ動画の関連サービスであるニコニ・コモンズでは、イラストも含めた素材の親子関係を記述するコンテンツツリーという機能が実装されている。これに

よりある作品の元になった作品や、ある作品を元にした他の作品を参照することができる。ただし登録は子作品の投稿者が手動で行わなければならないという制約があるため、コンテンツツリーが漏れなく網羅されている作品は限られている。



図 2.8: ニコニ・コモンズによって可視化されるコンテンツ間の親子関係

2.6 テキストの進化

手書きのメモやイラストと同様に紙の上で記録されていたテキストは計算機の登場により以下のような機能を獲得した。

- 編集支援機能
コピーやペースト、Undo や Redo 等の便利な機能によって簡単に文章が書けるようになった。
- ハイパーリンク
異なる文書への参照をハイパーリンクによって記述でき、素早く関連文書を参照できるようになった。
- ハイパーテキスト
ハイパーリンクやマルチメディアを埋め込めるハイパーテキストの登場で、よりリッチなコンテンツを作成できるようになった。

これにより参照・管理・再利用が難しいという問題点が解決されただけでなく、新しい活用法が発明されることとなった。

2.6.1 Wiki システム

ward らにより開発された Wiki[9] はハイパーリンクやマルチメディアを含んだハイパーテキストを手軽に作成・編集できるインターフェースを備えたアプリケーションである。複数人が共同で編集することも可能なのでコラボレーションツールとしても活用することができる。Wikipedia⁴はよく知られた Wiki の実装例の一つである。

(1) Scrapbox

Scrapbox⁵はGyazz[18]をベースに開発され、Nota⁶社によって運営されている Wiki システムで、他のシステムには無い以下のような特徴的な機能を備えている。

- シンプルな記法と WYSIWYG エディタ
Scrapbox ではページ間のリンクや、外部リンクを含んだ画像や動画、音声等のメディアを [] を基本としたシンプルな記法で記述することができる。
- 関連ページの表示機能
Scrapbox ではページの下部に
 - 別ページへのリンク
 - 別ページからのリンク
 - リンク先ページがリンクしているページ等のリンクに基づいた関連ページを表示する機能を備えている。Scrapboxにおいてコンテンツとなるページは全てフラットに扱われるが、ページ間リンクを元に関連する情報が自動的に推薦されるため、管理や分類を気にすることなくコンテンツを参照可能な状態にしておくことができる。

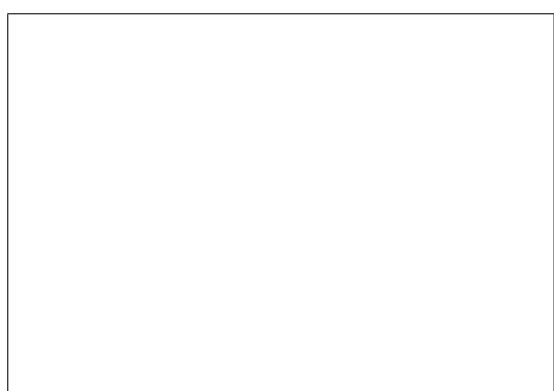


図 2.9: Scrapbox の画面



図 2.10: 関連ページリスト

⁴<https://www.wikipedia.org/>

⁵<http://scrapbox.io/>

⁶<https://www.notainc.com/ja>

2.7 手書きメモ・イラストの問題点

ハイパーテキストとは対照的に手書きメモ・イラストを表現する画像ファイルフォーマットが抱える制約によって、計算機が普及した現在でも以下のような問題が解決されていない。

- 参照や管理が面倒

手書きのメモ・イラストを参照可能な状態にするために、ファイル名や分類先のフォルダやラベリングするタグ等を工夫して管理しなければならない。

- 再利用が難しい

手書きメモをメモアプリ以外から参照する、別の手書きメモをリンクさせる等の用途が既存の手書きメモ・イラストを扱うシステムではそもそも想定されていないため、手書きメモを再利用する手段は大きく限定されている。

2.8 まとめ

手書きメモ・イラストは広く浸透した情報の記録手法であるものの、紙というフォーマットの制約によって使い勝手が制限されている。一方で計算機上で手書きメモ・イラストの作成や管理を行うシステムが広く利用されているが、画像ファイルフォーマットの制約からこれらは紙の手書きメモの利用形態を再現したにとどまり、参照や管理、再利用が難しいという本質的な問題は解決されていない。次章では上記のような問題点を解決するべく、これまでの手書きメモ・イラストの在り方にとらわれない次世代のフォーマット「ハイパーイラスト」と、それらを容易に管理・再利用できるシステム「手書きベース Wiki」を提案する。

第3章 設計

本章ではハイパーイラストと手書きベース Wiki の要件と設計について述べる。

3.1 要件

前章で示した画像ファイルフォーマットや手書きデータを扱う既存のツールの問題点を踏まえて、本システムの要件を整理する。

1. 簡単に手書きメモのメモやイラストが作成・編集できる
気軽に手書きメモ・イラストを取ることができ、また再編集性も可能である。
2. 作成した手書きのメモやイラストを簡単に参照したり、再利用したりできる
画像の内部に対してハイパーリンクを設定でき、関連する画像を参照することができる。

これらの要件を満たすシステムは次世代の画像ファイルフォーマットであるハイパーイラストと、その作成・編集と管理をサポートする手書きベース Wiki の組み合わせによって実現可能である。

3.2 ハイパーイラスト

本研究ではハイパーテキストの特徴を取り入れることによって既存の画像ファイルフォーマットの問題点を解決するハイパーイラストを提案する。

3.2.1 ハイパーイラストの定義

- テキストではなく手書きによって記述される
HTML 等のハイパーテキストは記法に基づいたテキストによってマークアップされるが、ハイパーイラストはタッチやスタイルスから取得される手書きデータによってその内容が記述される。
- 内部の任意の要素にハイパーリンクが埋め込むことができる
ハイパーテキストがその内部に他の文書への参照を示すハイパーリンクを埋め込むことができるよう、ハイパーイラストは内部にハイパーリンクを埋め込むことができる。

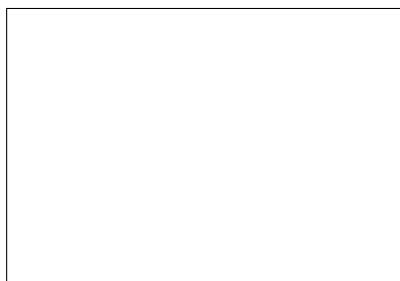


図 3.1: ハイパーイラストの概念図

3.2.2 ハイパーイラストの仕様

本研究におけるハイパーイラストは SVG[2] というフォーマットをベースとしている。SVG は XML¹をベースとしており、他の画像ファイルフォーマットにはない、ハイパーイラストに適した以下のような特徴を持つ。

- グラフィカルな表現を前提に設計されている

SVG には曲線等を表現する Path 要素や閉じた図形を表現する Polygon 要素等の仕様が標準で備わっている。スタイルスから得られるデータを Path 要素の属性として定義することで手書きのストロークを表現することができる。(例えばソースコード 3.1 で図 3.2 のように表現できる)

- 構造を保持できる

ラスターイメージと異なり SVG の実体は構造化されたテキストファイルであり、線分や点はピクセルではなく独立した要素として記述される。これにより書き順や編集履歴等の構造も保持され、再編集性が高い。

- ハイパーリンクを埋め込める

SVG では XLink²形式のハイパーリンクを任意の要素に埋め込むことができるため画像の中の個別の要素に対して複数のリンクを定義することができる。

- Web 標準の技術である

SVG は特定の企業の製品ではなく、その仕様は全て公開されている。また作成や表示に特別なソフトウェアを必要とせず、ブラウザのみで閲覧することができる。

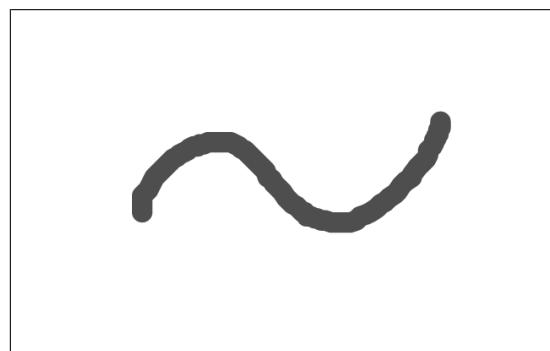


図 3.2: SVG 上で表現される手書きストローク

ソースコード 3.1: 図 3.2 の実体

```
1 <path stroke-linejoin="round" stroke-linecap="round" stroke="#585858"
2   stroke-width="10" class="" pointer-events="auto" fill="rgba(0,0,0,0)"
3   id="1072-528" d="M_919_564_L_919_563_L_919_562_L_919_560_L_919_559_
..._L_1064_520"></path>
```

¹<https://www.w3.org/XML/>

²<https://www.w3.org/TR/xlink/>

3.3 手書きベース Wiki

ハイパーイラストをコンテンツとする Wiki システムである手書きベース Wiki も提案する。

3.3.1 手書きベース Wiki の定義

- ハイパーイラストを作成・編集できる
タッチやスタイルスに対応したエディタを備え、手書きによってハイパーイラストを手軽に作成・編集することができる。
- ハイパーイラストにハイパーリンクを追加できる
作成したハイパーイラスト同士の相互リンクを簡単な操作で定義できる。
- 関連するハイパーイラストを参照できる
ハイパーイラスト同士のリンク関係を元に、関連するハイパーイラストを推薦し、一覧できるよう表示する。

この要件を満たす手書きベース Wiki のプロトタイプとして DrawWiki を開発した。

3.3.2 DrawWiki

DrawWiki は本研究において開発した、手書きベース Wiki のコンセプトを元にしたプロトタイプとなるアプリケーションである。(図 3.3) 本章ではその主要な機能を使い方とともに解説する。

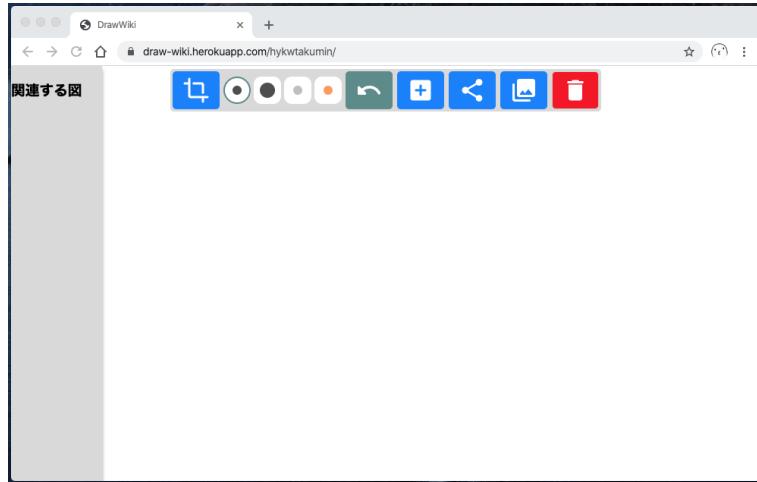


図 3.3: DrawWiki の初期画面

3.3.3 機能と使い方

(1) ハイパーイラストの作成・編集



図 3.4: ハイパーイラストの作成画面

画面中央部分に自由に手書きができるキャンバスが配置されており、ここに描いたものがハイパーイラストとして自動的に保存・アップロードされる。一度アップロードしたハイパーイラストには一意な URL が割り振られるため、Web を通じて他のユーザーが閲覧し、また編集することもできる。

(2) ブラシプリセット

メモやアイデアスケッチ等の用途であればさほど多くのブラシは要求されない。必要以上の機能はソフトウェアの使い方を習得する妨げとなりうるため、DrawWiki では金箱らによる Interactive Sketch[17]に基づいたコンパクトなブラシプリセット構成を採用している。このスケッチ技法は以下の 4 種類のブラシを用いる

1. 通常のペンで全体を描く
2. 影や質感をグレーで表現する
3. 輪郭を太い線で縁取る
4. 特徴的な部分、機能を有する箇所をキーカラーでハイライトする

これにより Wiki として活用する上で有利な以下のような特徴を持つ(図 3.5)。

- 太線で図のアウトラインが強調されるためサムネイル状態での視認性が確保される

- キーカラーによる強調表示でアイデアが伝わりやすくなる
- ブラシによって描き方がある程度統一され、スケッチ技量の巧拙を吸収できる



図 3.5: プリセットを用いて描かれた図

(3) ハイパーリンク埋め込み機能

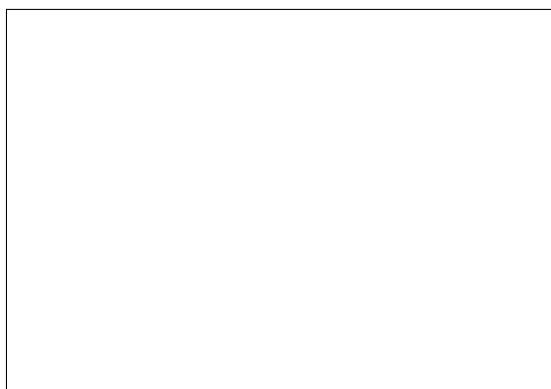


図 3.6: 範囲選択ツール

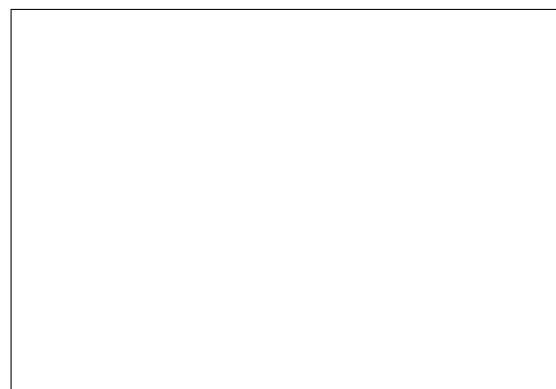


図 3.7: リンク埋め込み機能の操作画面

範囲選択ツールを用いてハイパーリンクを埋め込みたい要素を選択することができる。要素を選択して”他の図とリンクボタン”を押すと今まで作成したハイパーイラストのリストが開き、その中からリンクさせたい図を選ぶとその図へのハイパーリンクが要素に埋め込まれる。

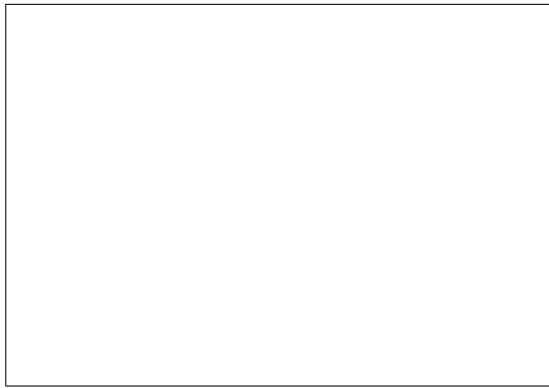


図 3.8: 関連イラストの表示機能

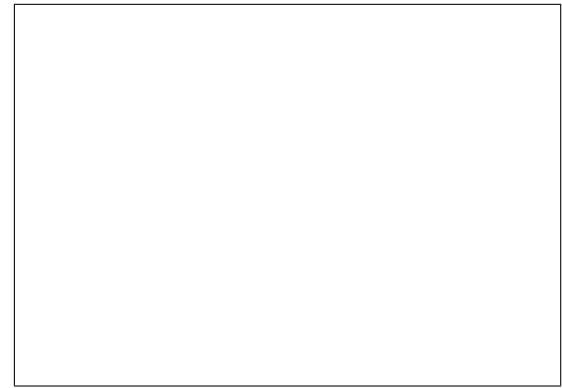


図 3.9: 関連イラストのモーダル

(4) 関連するハイパーイラストの表示機能

エディタの横に位置する”関連画像ビュー”には、

- 別のハイパーイラストへのリンク
- 別のハイパーイラストからのリンク

等のリンクに基づいた関連イラストのサムネイルが表示される(図 3.8)。またサムネイルを選択すると、当該ハイパーイラストがリンクしているハイパーイラストをリストするダイアログが表示される(図 3.9)。このようにリンクに基づいて関連するイラストを表示し、参照可能にする仕組みが備わっている。

(5) エキスポート・共有機能



図 3.10: エキスポート機能の操作画面



図 3.11: 埋め込まれたハイパーイラスト

エキスポート機能を利用すると、ハイパーイラスト単体(SVG ファイル)の URL を取得することができる。この SVG の URL は img 要素³や Object 要素⁴等のソースとして指定す

³<https://developer.mozilla.org/ja/docs/Web/HTML/Element/img>

⁴<https://developer.mozilla.org/ja/docs/Web/HTML/Element/object>

ることが可能で、図 3.11 のように他の Web サイトにも埋めこむことができる。

第4章 実装

本章では第3章で述べたシステムの設計を受け、DrawWikiの実装について述べる。

4.1 アプリケーション構成

DrawWiki は Web アプリケーションとして実装されており、HTML5 と SVG1.1 に準拠したブラウザがインストールされていれば、OS やデバイスに依存せず利用することができる。本アプリケーションの構成は以下の図の通りである。



図 4.1: アプリケーションの構成

4.2 クライアントサイド

実際にハイペーイラストを作成したり関連イラストを閲覧したりするクライアントサイドのプログラムは HTML¹と javascript²によって実装される。開発には Javascript にコンパイル可能な漸進的型付け言語 TypeScript³を用いている。

4.2.1 手書きデータの取得

DrawWikiにおいて指やスタイラスの操作から生じる座標等の手書きデータを PointerEvent API⁴によって取得している。この API は TouchEvent や MouseEvent と異なりタッチやマウスだけでなくスタイラスを含めたあらゆるユーザー入力を透過的に扱うことが可能で(図 4.2)、また主要なブラウザに全て実装されているためあらゆるプラットフォーム・デバイスから利用できる。

4.2.2 仮想 DOM の利用

DrawWiki は大量に発生する PointerEvent から動的に手書きストロークを生成し、ハイペーイラストを作成・描画するが、単に描画するだけでなく内部の任意の要素にハイペー

¹<https://developer.mozilla.org/ja/docs/Web/HTML>

²<https://developer.mozilla.org/ja/docs/Glossary/JavaScript>

³<https://www.typescriptlang.org/>

⁴<https://developer.mozilla.org/ja/docs/Web/API/PointerEvent>

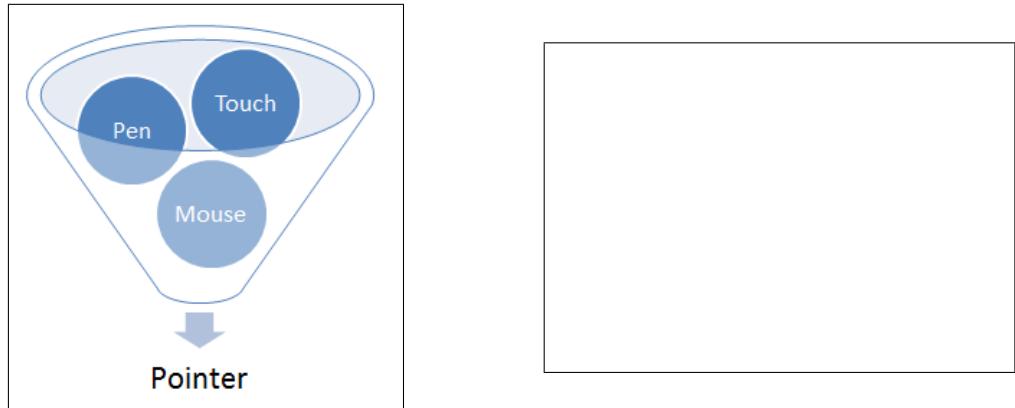


図 4.2: PointerEvent API の概要

図 4.3: 仮想 DOM のモデル

ンクを埋め込んだり、別のハイパーイラストをインポートしたり等の複雑な操作を行うことがある。例えばハイパーリンクを追加する際は以下のような処理を実行している。

1. 範囲選択された座標内にあるストロークを SVGGroup 要素で囲う
2. グループ化したストローク群をさらに SVGAnker 要素で囲う
3. SVGAnker 要素の href 属性にリンクさせるハイパーイラストの URL を代入する

ハイパーイラストはベースとなる SVG と同様にあらゆる要素が階層構造的に記述されており、そのようなツリーの操作は DOM インターフェース (appendChild や removeChild、insertBefore 等) を通じて行なうことが一般的である。

しかし例のような要素が階層構造内を横断するような複雑な処理を DOM インターフェースのみで行なうことはコードの複雑化と描画パフォーマンスの低下を招く。そこで DrawWiki では実際の DOM を操作するのではなく、仮想 DOM に対してのみ操作を行い、その差分を実際の DOM に Dispatch する手法を用いている。仮想 DOM を実際のハイパーイラストに反映する処理は View ライブリの React⁵を利用している。

4.2.3 debounce された更新処理

DrawWiki ではハイパーイラストが編集され変更が生じる度に自動でデータを上書き保存する仕様だが、画面上で指やスタイルスペンを動かしている間は常に PointerEvent が発生し続けているため、このまま変更をアップロードするとサーバー側の処理能力を超える頻度でリクエストを送信してしまう。そこで Event 発生後にタイマーをセットし、2 秒経過した段階でアップロード処理を行う debounce 機能を実装した。このタイマーは新しい Event が発生する度にリセットされるため、ペンを置いてしばらく、つまり最後の Event から 2 秒間新たな Event が発生しないことが確定してからアップロード処理が行われる。これによりア

⁵<https://ja.reactjs.org/>

プリケーションサーバーやアセットサーバーに過大な負荷を与えることなく更新処理を行うことができる。

4.2.4 リンク付要素の視覚的表現

関連イラスト表示にはリンクしているハイパーイラストのサムネイルが表示されるが、そのサムネイルを選択すると、そのハイパーイラストをリンクされている要素が変化し、対応関係が視覚的に表示される。ハイパーイラストのベースとなる SVG は CSS⁶を適用可能であり、この視覚効果も CSS と CSS @keyframes⁷を組み合わせることで実現している。

4.3 サーバーサイド

サーバーサイドはアプリケーションサーバーとアセットサーバーとで構成されている。

4.3.1 アプリケーションサーバー

DrawWiki は Node.js⁸ 上で動作する Web アプリケーションとして実装されている。HTTP リクエストを処理する Web アプリケーションフレームワークとして Express⁹を用い、そのホスティング環境として BaaS(Backend-as-a-Service) の一つである Heroku¹⁰を利用している。

(1) ハイパーイラストのアップロード処理

クライアントは作成したハイパーイラストを”multipart/form-data”形式でエンコードイングし、アプリケーションサーバーはそのデータを POST 通信で受け取る。受け取ったハイパーイラストをもとにメタデータを生成し、双方をアセットサーバーに送信することでアップロード処理を行う。

また既存のハイパーイラストや付随するメタデータの読み取り、更新、削除については REST 原則 [6] に基づいたルーティングによって処理する構成となっている。

(2) メタデータの構成

ハイパーイラストの本体とは別に、以下のようなメタ情報を管理するために DrawWiki ではメタデータファイルも扱う。

⁶<https://developer.mozilla.org/ja/docs/Web/CSS>

⁷<https://developer.mozilla.org/ja/docs/Web/CSS/@keyframes>

⁸<https://nodejs.org/>

⁹<https://expressjs.com/>

¹⁰<https://www.heroku.com/>

- ハイパーイラストの作成者
- 最終更新日時
- 引用している、またはインポートしているハイパーイラストのリスト

ソースコード 4.1: メタデータの概要

```

1  export type HyperIllust = {
2    id: string; //ハイパーイラストの ID
3    sourceKey: string; //ハイパーイラストの本体ファイルの Key
4    sourceURL: string; //ハイパーイラストの本体ファイルの URL
5    size?: number; //ファイルサイズ
6    linkedList?: string[]; //リンクされているハイパーイラストのリスト
7    linkedByList?: string[]; //このイラストをリンクしているハイパーイラストのリスト
8    importedList?: string[]; //インポートしたハイパーイラストのリスト
9    importedByList?: string[]; //このイラストをインポートするハイパーイラストのリスト
10   createdAt: DateLike; //作成日時
11   updatedAt: DateLike; //更新日時
12   owner: string; //作成者の名前
13 };

```

4.3.2 アセットサーバー

アプリケーションサーバーはステートレスなプロセスとして実行されるため、変数やファイル等を保存し永続化する仕組みをもたない。ファイルとしてのハイパーイラストやメタデータを保存するために DraWiki はアセットサーバーとしてクラウドストレージである AWS S3¹¹ を利用している。アップロードされたハイパーイラストは DrawWiki 以外の Web サイトからも閲覧・再利用できるように、ACL(Access-Control-List) を公開読み取り ("public-read") に設定している。またアセットサーバー自体もオリジン間リソース共有¹² を有効化している。

¹¹<https://aws.amazon.com/jp/s3/>

¹²<https://developer.mozilla.org/ja/docs/Web/HTTP/CORS>

第5章 應用例

本章では、手書きベース Wiki によって実現可能な應用例について述べる。

5.1 ハイパーイラストを活用したメモ

手書きメモ・イラストの内部に別のイラストへのリンクが埋め込めるというハイパーイラストの特徴を活かし、次のような活用を行った

5.1.1 ソフトウェアのプロトタイピング

ソフトウェアの画面や挙動を設計する際に、柔軟な表現が可能な手書きスケッチを用いることが有効なデザインプラクティスとして知られている。DrawWikiを開発する際も、機能や画面全体のデザインを考える際にDrawWikiを用いてスケッチを行いながら検討した。リンクが埋め込まれた要素をクリックするとリンク先ハイパーイラストが開くことから、これを画面遷移に見立てることにより図??のようなナビゲーションを伴ったソフトウェアのモックアップを手書きスケッチによって作成することができる。

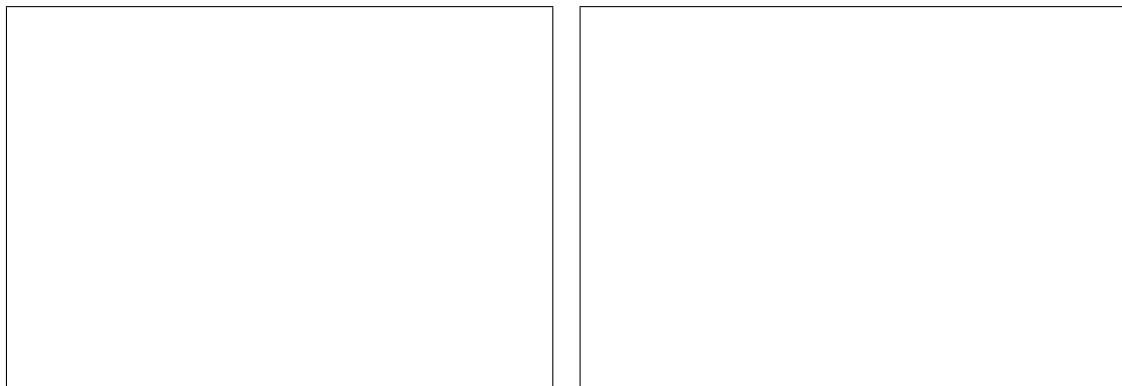


図 5.1: DrawWik を用いて作成された Drawwiki のモックアップ

図 5.2: アクションによってポップアップするダイアログの再現

5.1.2 自作の整備メモ

自動車等の整備を行う際は、製品マニュアルを熟読することで構造を理解し、交換すべき部品をパーツリストで確認し、さらに取り付けに必要なワッシャ・ボルトのサイズやそれらの適正な締め付けトルクを念頭に置いた上で行う必要があるが、それらの情報は一般的に異なるページや冊子に分かれて記載されているため、参照が大変である。ハイパーイラストの持つ機能によって製品の構造をわかりやすく描いた上で、必要となる部品の情報や、組み立てに要する注意等をリンクさせ必要な情報にすばやくアクセスできる、視認性に優れたメモの作成が可能である。

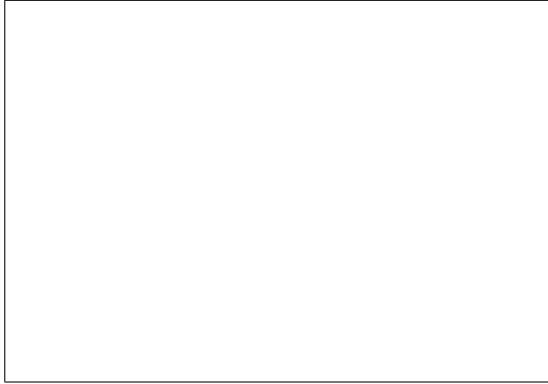


図 5.3: DrawWiki による自作整備メモ

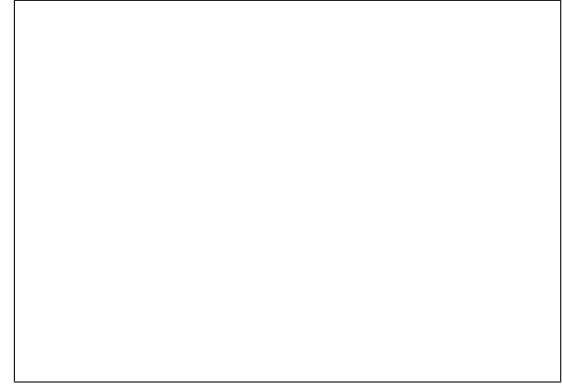


図 5.4: 関連情報を表示した画面

これらのメモは全く異なる分野を取り扱ったものであり、従来のメモアプリケーションでは異なるフォルダに格納する、タグやラベル等を貼り付ける等の手段で分類されるべきものだったが、ハイパーイラスト内のリンク情報に基づいて関連イラストが表示される DrawWiki では、情報の整理や適切な管理を特別心かけなくともリンクを辿ることで目的のハイパーイラストを参照することが可能である。

5.2 ナレッジの共有と共同編集

DrawWiki によって作成されたハイパーイラストには URL が割り振られているため、画像として他の Web サイト内に埋めこむことも可能である。この状態でもリンク情報は失われずファイル内に内包されているため、リンクされている関連画像も含めたナレッジとして共有することができる。さらに他のユーザーによる内容の追記や、別のハイパーイラストをリンクさせるといった操作にも対応しているため手書きメモを取るような気軽さはそのままに Wiki のようなコラボレーションを行うことができる。

5.3 手書きによる Web ページ・Wiki の作成

ハイパーイラストはそれ自体がハイパーリンクを内包したハイパーテキストであるため手書きという簡単な操作のみで Web ページを作り、公開することができる。通常のテキストベースの Wiki はテキストによって内容を記述することを前提としているが、テキスト入力に起因する以下のような問題点がある。

1. 効率的なテキスト入力はキーボード等の専用ハードウェアが必要であり、これらのデバイスを備えていないスマートフォンやタブレット等のハードウェアで便利に入力することができない
2. キーボードの操作にはタッチタイピング等の技術に習熟している必要があるため、利用者にはある程度の技能が要求されてしまう。

3. Wiki コンテンツの記述には専用の記法に倣う必要がある。例えばハイパーテキストを記述する上で代表的な言語である HTML では、テキストへのハイパーリンクの埋め込みを以下のように定義する。

ソースコード 5.1: html におけるハイパーリンクの定義

```
1 <a href="https://example.com">HyperLink</a>
```

また HTML へと変換できるプレーンテキストのフォーマットとして広く普及している Markdown では、同様の構造を以下のように記述する。

ソースコード 5.2: html におけるハイパーリンクの定義

```
1 [HyperLink]("https://example.com")
```

上記の記法とは別に独自の記法を採用している Wiki も存在するため、その各々の記法も利用者は網羅していなければならぬ。

(1) についてはタッチパネルさえあれば手書き入力が可能であるため、スマートフォンやタブレット等のキーボードを持たないデバイスからでも利用することができる。(2) に関しても、手書きはタイピングが登場する以前から存在する馴染み深いベーシックな入力方法であり、タッチタイピング等の技能を持たないが手書きはできる人でも利用することができる。ハイパーリンクの定義も作成も手書きとシンプルな操作で完結するため(3) のような記法を覚える必要もない。

5.4 まとめ

本章では、本システムによって実現可能な応用例について述べた。ハイパーイラストと手書きベース Wiki の組み合わせによって既存の手書きメモ・イラストの問題点を解決でき、またテキストベース Wiki に対しても優れた点があることがわかった。本章で述べた応用例に限らず、様々な応用が可能と考えられる。

第6章 関連研究

本章では関連研究を紹介し、それらの特徴や本研究との関連性について示す。

メモを含めて手書きデータの活用を目的とする先行研究は数多く存在する。

6.1 主要な研究領域

手書きメモ・イラストに関する主要な研究領域を解説する。

6.2 手書きメモ支援システム

タブレットやPDA等のデバイスを用いて手書きメモの作成や運用を効率化する研究が行われている。

6.2.1 Notepal

Davisらが提案するNotepal[3](図6.1)はPalm Pilot等のPDAを用いて手書きメモをグループで共有可能にするシステムである。他のマシンからもメモを閲覧できるようにクレードルでホストマシンと接続した際にデータを共有スペースにアップロードする。これにより個人のメモを他のメンバーが参照し、組織的に活用することを可能にした。またPDAの限られた画面面積では入力しづらいケースに対応するためデジタルペンによって描かれた紙のデータも共有することができる。一方で手書きストロークの中にリンクを埋めこむ等の機能はないためメモ同士をリンクさせるといった活用はできない。

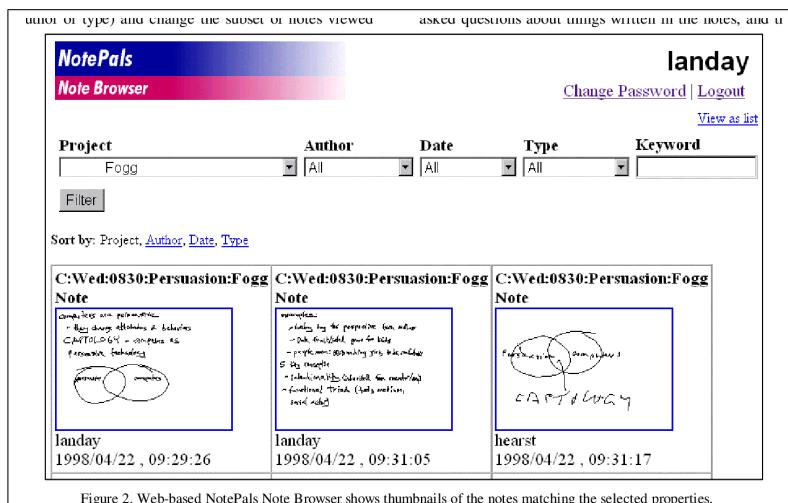


図 6.1: Notepal の画面

6.2.2 InkSeine

Hinckleyらによって開発されたInkSeine[7](図6.2)はペンインターフェースを備えたタブレットPCを主眼に置いた電子スクラップブックシステムである。キーボードは使わずにペ

ンのみで操作が完結するよう UI が設計されており、テキスト入力も手書き文字認識によって行う。また入力した手書き文字をファイルや Web ページの検索に活用したり、検索結果のスナップショットをハイパーリンクと共にキャンバスに貼り付ける等の機能を備えている。一方でスクラップブックをタブレットデバイス上で再現することを目的としているため、共同編集等の活用法は想定されていない。

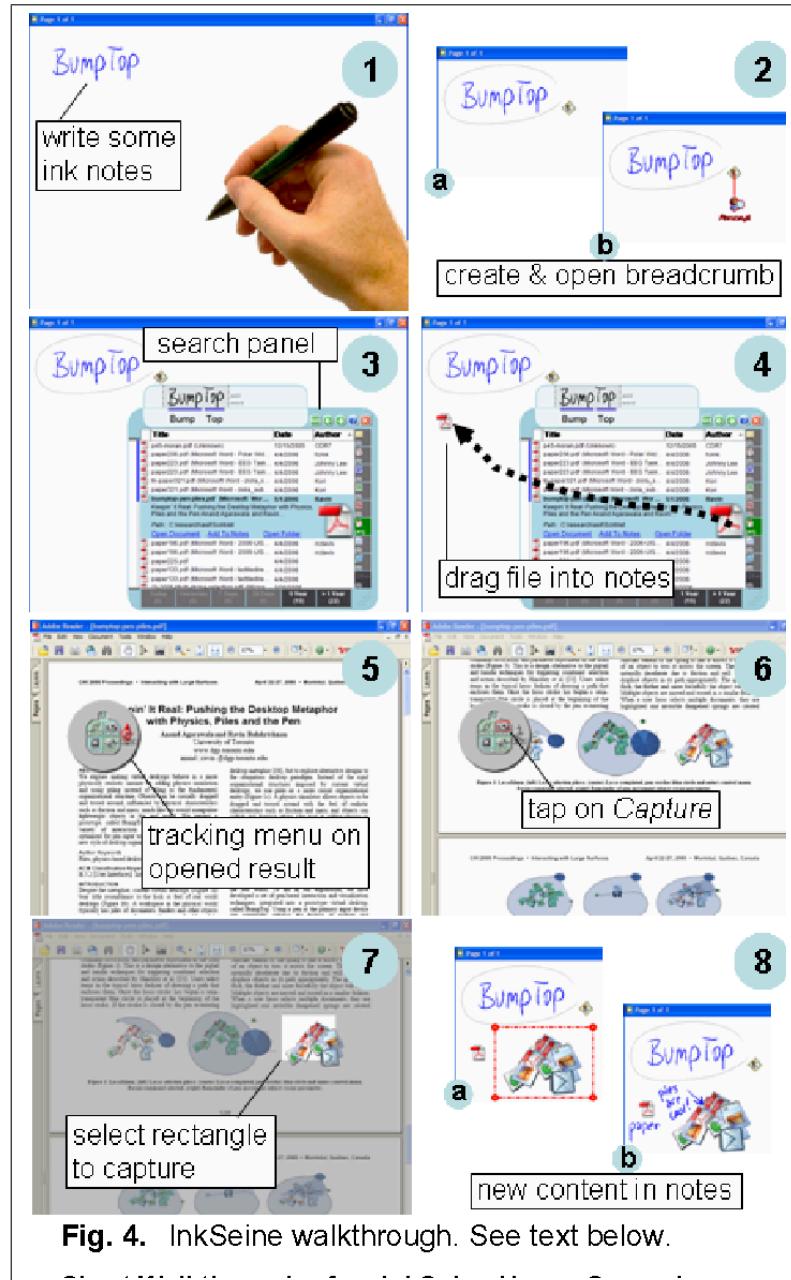


図 6.2: InkSeine の操作画面

6.3 デザイン・プロトタイピングツール

アプリケーションや Web サイトの UI を設計する際スケッチがよく利用される [12]。スケッチとしての手書きデータをデザインやプロトタイピングに取り入れる試みとして以下の研究が挙げられる。

6.3.1 DENIM

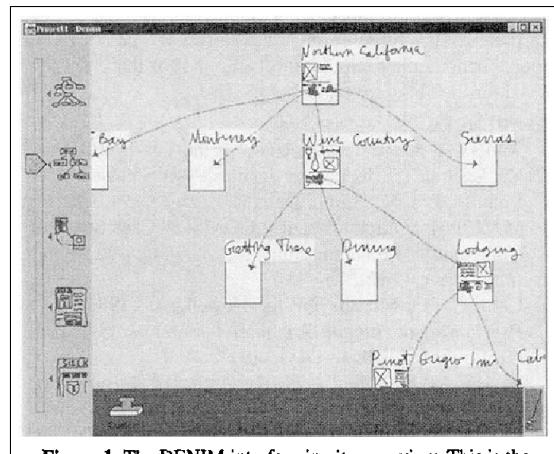


Figure 6.3 The DENIM interface in site map view. This is the

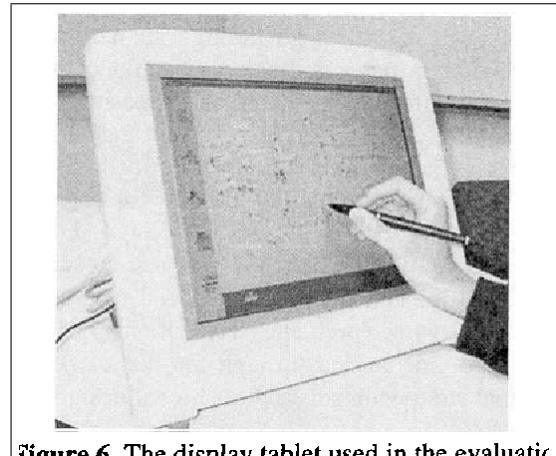


Figure 6.4 The display tablet used in the evaluati

図 6.3: DENIM の画面

図 6.4: 操作を行うタブレット機器

Lin はタブレットからのスケッチを元に Web サイトのデザインやナビゲーションの設計の支援を行うシステム DENIM を開発した [11](図 6.3)。図 6.4 のようなタブレット機器から操作できるように設計されている。手書き入力だけでなく Web サイトのディテールから大まかな画面遷移に至るまでをカバーするためにスケッチを行うキャンバスにはズーミングインターフェースが備わっている。

6.3.2 SILK

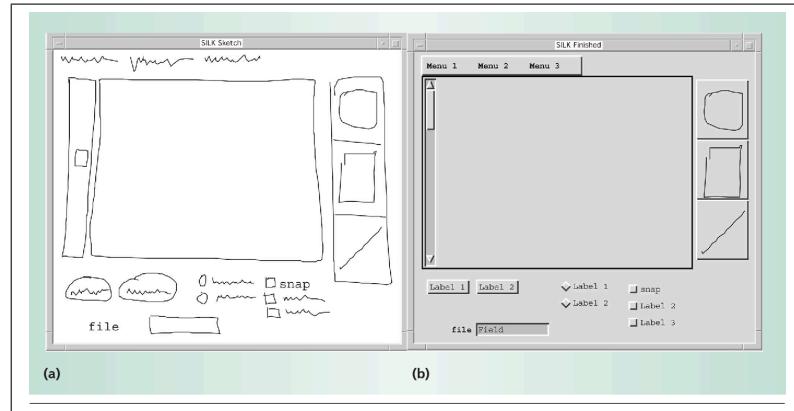


図 6.5: SILK の画面

Landay らはスタイラスから入力した手書きデータから実際に動作する GUI を構築する SILK (Sketching Interfaces Like Krazy) というシステムを実装した [8](図 6.5)。GUI を設計する際は紙にアイデアをスケッチすることが一般的だったがこのシステムでは手書きのスケッチを元に実際に動作するプロトタイプを作成することができる。

6.3.3 IntaractivePaper

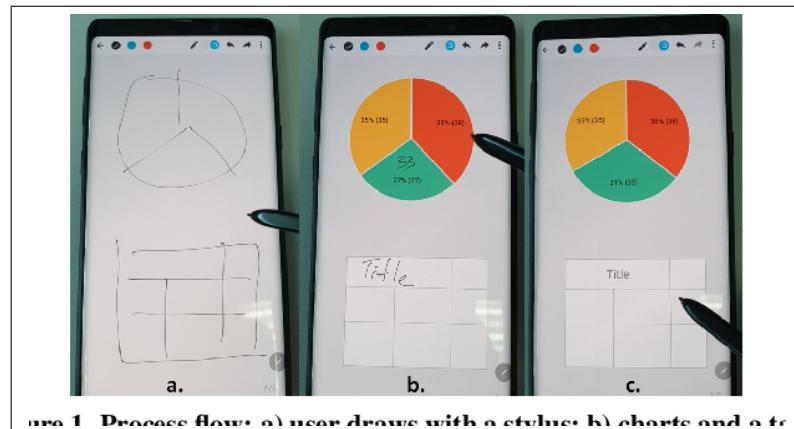


図 6.6: IntaractivePaper の画面

Zhelezniakov らによる IntaractivePaper[16] ではラフな手書きストロークからパイヤートやテーブル等の要素に変換する機能を備えている。またモバイル機器から利用できるよう認識フローや選択インターフェースが再検討されている(図 6.6)。

6.4 手書きデータを利用した Active Reading やコラボレーション

Active Reading や共同作業を行う場合にも手書き入力は頻繁に活用される。

6.4.1 XLibris

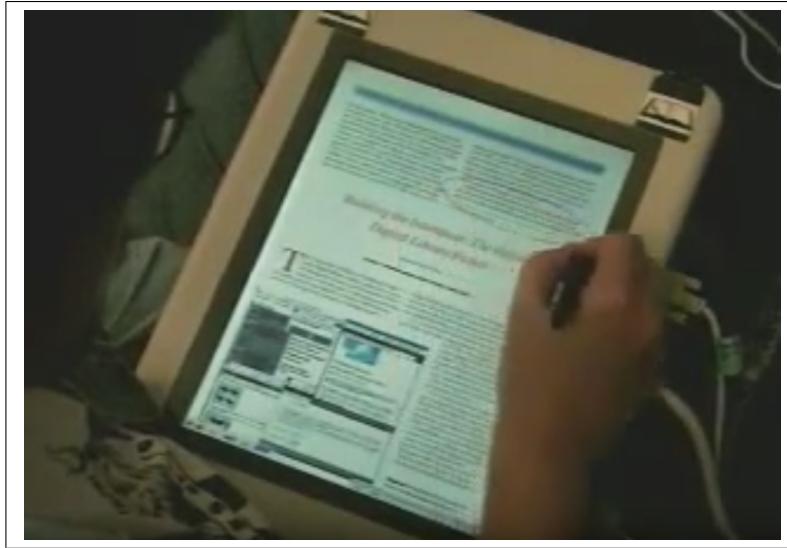


図 6.7: 操作中の XLibris

Price らによる XLibris[14] はタブレットデバイス上で Active Reading を行うことを目的としたシステムである。手書きによる注釈やハイライトの入力や、入力したページを一覧画面から参照する機能等を備えている(図 6.7)。Active Reading をタブレットデバイスで再現することを主眼に置いているため、文書の脇に追記する注釈以上の、自在な手書きメモやイラストを書き加えるという用途は想定されていない。

6.4.2 SketchComm

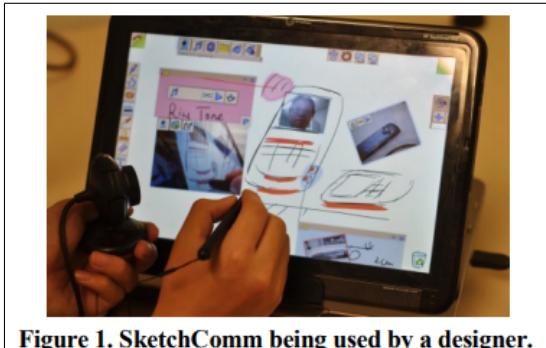


Figure 1. SketchComm being used by a designer.

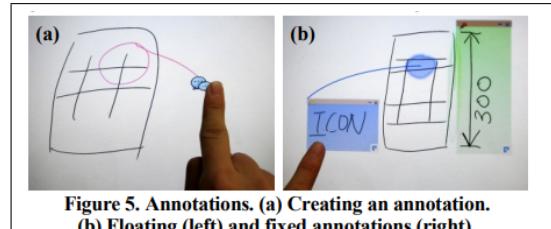


Figure 5. Annotations. (a) Creating an annotation.
(b) Floating (left) and fixed annotations (right).

図 6.9:

図 6.8:

Li らによる SketchComm[10] はアイデアに関するスケッチが非同期的にレビューされることを念頭に置いて設計されたシステムである(図 6.8)。対面によるリアルタイムのコミュニケーションと異なりコンテキストが欠落しないよう、注釈やコメント、音声データ等をスケッチの内部に埋め込むことができる(図 6.9)。SketchComm はコラボレーションツールではあるものの、Creator と Reviewer という役割を持ったユーザー同士という特殊な状況に特化しているため Wiki のようなカジュアルな共同編集は想定されていない。

第7章 考察

本章では、手書きベース Wiki の自身の運用経験や利用者の意見をまとめ、諸問題や研究の重要性・発展性について述べる。

7.1 評価

手書きベース Wiki のプロトタイプとなる「DrawWiki」は 2019 年 10 月に Web アプリケーションとして公開した¹。また ORF2019²にて DrawWiki の展示発表を行った。本節では、DrawWiki の筆者の運用経験や展示発表で得られた感想、ユーザーからのフィードバックをまとめることとする。

7.1.1 筆者の運用経験

主に個人用のメモとして活用しつつ、DrawWiki の開発と並行して 3ヶ月間利用した。

ハイパーイラストの有用性 第 5 章で示したように、ソフトウェアのモックアップや自作の整備メモ等の従来のメモアプリでは表現できなかったハイパーイラストならではのコンテンツを作成することができた。ハイパーイラストの作成そのものは Inkscape³等の SVG を編集できるツールを用いれば可能であるが、

- 手書きの画像データを用意し
- 紐付けたい他の画像データをクラウドストレージにアップロードし
- その URL を埋め込んだ上で SVG として保存する

という一連の作業を全て手作業で行う必要があり、気軽に作成することはできなかった。DrawWiki によって簡単な操作でイラストの内部へのリンクの追加が可能になったため、手書きメモというカジュアルな用途でもハイパーイラストを作成し、その恩恵を受けられるようになった。

リンクに基づく優れた参照性 DrawWiki で作成したメモ・イラストにはリンクこそ埋め込まれているものの、フォルダやタグ・ラベル等のメモを分類し管理するための機能は存在しない。全てのメモ・イラストがフラットに置かれているものの、リンク情報に基づいて関連するメモ・イラストが推薦されるため、参照に困ることはなかった。既存のメモアプリではメモを描く前に

- どのようなファイル名にするか?
- どのようなフォルダに分類するのか?
- どのようなタグ・ラベルを付けるのか?

等を意識する必要があったが、そのような管理の工夫をせずとも、リンクによるシンプルなナビゲーションがあれば目的のメモ・イラストを参照できることがわかった。

¹<https://draw-wiki.herokuapp.com/>

²<https://orf.sfc.keio.ac.jp/2019/>

³<https://inkscape.org/>

7.1.2 意見・感想

第2.7節で述べた手書きメモ・イラストの問題点について多くの同意が得られた。多くのユーザーが同様の問題を抱えつつも、本質的に解決する手段・方法が無かつたため各々が運用上の工夫・苦労を強いられていた。その他に以下のような感想や意見が寄せられた。

関連資料のナレッジ化 デザインの課題では多数のスケッチを描く場面があるが、それらは紙で管理されているため管理しなければ散逸してしまい、また振り返るために参照するのも大変である。これを既存のメモアプリケーションに置き換えると紙と同様の参照・管理の不便さが解決しないが、スケッチにリンクを追加することで関連する作品を漏れなく参照することができ、生きたナレッジとして活用できるのではないかという意見を得た。

手書きで作成できる Wiki タイピングやPCの操作が得意でないためにテキストベースWikiを使う機会がなかった人からも、手書きのメモを描いてそれが繋がるのであれば自分でもコンテンツを作れそうという意見があった。第5.3節で示したように、手軽にハイペーイラストが作成でき、それらをコンテンツとするWikiがあれば、高度なスキルがなくてもカジュアルにWikiに参加することが可能であると考えられる。

7.1.3 問題点・要望

以下のような問題点が明らかになった。

1. テキスト検索によってメモを参照したい

メモの内部にある手書きの文字を対象に、テキスト検索によって参照できる機能がほしいという要望があった。現状のDrawWikiにおいて文字として描かれた手書きストロークを特別に区別・認識していないため、テキストによって検索する機能を備えていない。手書きメモであってもテキストによって全文検索したい

2. 編集履歴が分かりづらい

基本的にハイペーイラストはURLを通じていつでも、誰でも編集可能であるが、どのように編集されたのか、どのような内容が追記されたのかを判断しづらい場合がある。また手書きストロークが誰によって追加されたのかを視覚的に表示する機能を備えていないため、共同編集を行った際に誰による編集なのかを把握しづらいという指摘があった。

3. リンクを追加する際の補助が欲しい

他のメモへのハイパリンクを埋め込む際に、どのメモを埋め込むかはモーダルによって選ぶDrawWikiではある要素に対して別のメモ・イラストをリンクさせたる際に、今までに作成したメモ・イラストの一覧から選択するという操作になっている。この一覧は時系列順や引用数/被引用数によってソート・絞り込みを行うことが可能だが、さらに進んでメモを取っている最中にその時点での手書きストロークに類似する過去

のメモやイラストを候補として推薦する仕組みがあれば、よりスムーズなリンクの埋めこみやイラストのインポートが可能になるという意見があった。

7.2 考察

7.2.1 設計指針の妥当性

本システムは既存のメモアプリケーションとは異なる利用形態を持つが、実際に利用した、またはデモを体験したユーザーからは概ね好意的に受け止められたため、ハイパーリンクとWikiの組み合わせという設計は適切であったと言える。本システムをベースに様々な改良を重ねることで、より優れた手書きメモ・イラストの利用環境の実現や応用が可能であると考えられる。

7.2.2 解決すべき課題

1. テキスト検索による参照

DrawWikiではスタイルスから得られる手書きデータを元にストロークを生成しているため、オンライン手書き文字認識を応用することでメモ内のテキストを認識し、クエリとして検索・参照する機能は技術的に実装可能である。テキスト検索機能を備えることで、よりメモを参照しやすい状態で運用することができると考えられる。

2. 編集履歴の可視化

DrawWikiで管理されるハイペーイラストでは編集に関する履歴情報を保持している。これらの情報を元に、編集者によってストロークの色や形状に変更を加えたり、新しく編集された部分について強調表示したり等の視覚表現を追加する機能があればより便利に共同編集を行うことが可能になると考えられる。

3. ”手書き入力補完”によるリンク追加処理の支援

スケッチしている図形を認識したり[13][15]、それをクエリとして画像検索を行う手法は数多く存在する[4][5][1]。これらを応用することで描画中の手書きストロークを元に過去に作成したメモ・イラストの中から類似したものを検索し、候補として提案する”手書き入力補完”的な機能を実装することができ、よりスムーズに手書きメモ・イラスト同士をリンクさせることができるようになる。

7.2.3 手書きメモ・イラストの問題点の検証

本システムにおいて第2.7節で述べた問題点が克服されているかどうかを検証する。

- 参照や管理が面倒

手書きメモ・イラストのリンク情報に基づいて関連イラストを推薦する機能により、管理に労力を割くことなく目的のメモ・イラストを参照できるようになった。

- 再利用が難しい

手書きメモやイラスト同士をリンクさせたり、他の Web サイトに埋め込んだりといった再利用が可能になった。

以上のように、第 2.7 節で述べた問題点がハイパーイラストと手書きベース Wiki によって解決した。

第8章 結論

本章では本研究を総括する。

8.1 研究の成果

本研究では、ハイパーリンクを内蔵した次世代の手書きデータ記述形式「ハイパーイラスト」と、それをコンテンツとして扱う Wiki システム「手書きベース Wiki」の提案を行った。まず第 2 章において、既存の手書きメモ・イラストの問題点をテキストの進化と比較しながら分析した。既存システムの現状をとりあげ、計算機が普及した現在も根本的に解決されていないことを示した。第 3 章では、第 2 章で述べた手書きメモの問題点に対する解決方法を提案した。また、それに基づき本研究で開発した「ハイパーイラスト」「手書きベース Wiki」の基本構成と使い方について述べた。第 4 章では、「手書きベース Wiki」のアプリケーション構成と詳細な実装について述べた。第 5 章では、「手書きベース Wiki」によって実現可能な応用例について述べた。第 6 章では、本研究に関連する研究を紹介し、それぞれのアプローチの特徴と問題点を分析した。第 7 章では、筆者による運用経験やユーザーからのフィードバックをもとに本研究の有効性と問題点を分析した。

8.2 総括

本研究では手書きのメモやイラストの中に自在にハイパーリンクを埋め込めるフォーマット「ハイパーイラスト」と、それらをナレッジとして扱える「手書きベース Wiki」の開発を行った。「手書きベース Wiki」はハイパーリンクや Wiki 等の技術の組み合わせによって既存の手書きのメモ・イラスト問題点を克服し、新しい活用法を実現した。今後は第 7 章で述べた問題点を受け、システムを改善していく。

謝辞

慶應義塾大学環境情報学部 増井俊之教授には学部から5年間の長きに渡りご指導を賜りました。深く感謝いたします。また、本研究の副査としてご意見、ご助言を頂きました中西泰人教授、武田圭史教授に感謝いたします。また自身の研究について幅広い議論をしていたいた政策・メディア研究科博士課程の田中優氏、大和比呂志氏を初め、様々な形でアドバイスをくださった増井俊之研究会OB諸氏に感謝いたします。

2020年1月 慶應義塾大学 政策・メディア研究科 修士2年 早川匠

参考文献

- [1] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: internet image montage. In *SIGGRAPH 2009*, 2009.
- [2] Dean Jackson Chris Lilley. Scalable vector graphics (svg). 10 2004.
- [3] Richard C. Davis, James A. Landay, Victor Chen, Jonathan Huang, and Rebecca B. Lee. Notepals: lightweight note sharing by the group, for the group. In *CHI '99*, 1999.
- [4] Mathias Eitz, Ronald Richter, Tammy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Trans. Graph.*, Vol. 31, pp. 31:1–31:10, 2012.
- [5] Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tammy Boubekeur, and Marc Alexa. Photosketcher: Interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, Vol. 31, pp. 56–66, 2011.
- [6] Roy T. Fielding. Architectural styles and the design of network-based software architectures (chapter 5). 2000.
- [7] Ken Hinckley, Shengdong Zhao, Raman Sarin, Patrick Baudisch, Edward Cutrell, Michael Shilman, and Desney S. Tan. Inkseine: In situ search for active note taking. In *CHI*, 2007.
- [8] James A. Landay and Brad A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, Vol. 34, pp. 56–64, 2001.
- [9] Bo Leuf and Ward Cunningham. The wiki way: Quick collaboration on the web. 2001.
- [10] Guang Li, Xiang Cao, Sergio Paolantonio, and Feng Tian. Sketchcomm: a tool to support rich and flexible asynchronous communication of early design ideas. In *CSCW '12*, 2012.
- [11] James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. Denim: finding a tighter fit between tools and practice for web site design. In *CHI '00*, 2000.

- [12] Mark W. Newman and James A. Landay. Sitemaps, storyboards, and specifications: a sketch of web site design practice. In *DIS '00*, 2000.
- [13] Matthew J. Notowidigdo and Robert William Clinton Miller. Off-line sketch interpretation. In *AAAI 2004*, 2004.
- [14] Morgan N. Price, Bill N. Schilit, and Gene Golovchinsky. Xlibris: the active reading machine. In *CHI '98*, 1998.
- [15] Jacob O. Wobbrock, Andrew D. Wilson, and Yang D. Li. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In *UIST '07*, 2007.
- [16] Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko, and Yevhenii Yakishyn. Interactivepaper: Minimalism in document editing ui through the handwriting prism. In *UIST '19*, 2019.
- [17] 淳一金箱, 直蛭田, 克彦原田, 俊介高尾, 裕行佐竹, ギブソンジェームズ, 亨赤羽. 相互作用を喚起するアイデアスケッチ手法 : Interactive sketch の提案. 日本デザイン学会研究発表大会概要集, Vol. 58, pp. 14–14, 2011.
- [18] 増井俊之. Gyazz-柔軟で強力な万人のための wiki システム. 第 52 回プログラミングシンポジウム予稿集, 第 2011 卷, pp. 43–50, jan 2011.