筑波大学大学院博士課程

システム情報工学研究科修士論文

グラフィックデザインの実環境における プレビュー及び編集を可能にする ARインタフェース

岩谷 明

修士（工学）

(コンピュータサイエンス専攻)

指導教員　高橋 伸

2016年3月

概要

ポスター等の印刷物のグラフィックデザイン制作では，コンピュータ上で動作する Desktop Publishing (DTP) ソフトウェア及び液晶ディスプレイを用いて作業し，制作物を印刷，その後に納品や掲示を行う．しかし，製作中のデザインが実際に掲示する環境ではどう見えるかを確認するためには，試し刷りなどの煩雑な作業が必要となる．そこで，本研究では，実環境におけるグラフィックデザインのプレビュー及び編集を，拡張現実技術を用いて可能にする．まず，拡張現実 (AR) によるデザインのプレビュー・編集にてユーザーに行われるタスクを提案する．そして，各タスクに対するインタフェースを備えたシステムとして，まず，実環境にすでに掲示されているポスターの編集を可能にするプロトタイプシステムを開発した．さらに，より複雑なデザイン作業に対応した，主となるデザイン制作を行うデスクトップコンピュータ用アプリケーションと，仮想ポスターをプレビュー・編集できるスマートフォンアプリケーションが連携するシステムの開発を行った．また，実環境の照明色を，仮想物体にリアルタイムに反映する簡易的な手法を提案し，実環境に印刷物を掲示した際の見え方に近いプレビューを可能とする．そして，前述のインタフェースを含んだモバイル端末およびコンピュータからなる提案システムの各種機能およびそれらの実装方法について詳細に述べる．最後に，システムの評価実験においては，提案した照明色反映手法の有用性を確認し，さらに被験者による評価実験にて仮想ポスターによるデザインシステムがグラフィックデザイン制作において有用であることを示した．

目 次

図 目 次

第1章　序論

1.1　背景

ポスターやフライヤーといった印刷物のデザインでは，Adobe Illustrator に代表される Desktop Publishing (DTP) ソフトウェアが用いられる．各種の DTP ソフトウェアでは，ディスプレイに表示される内容を印刷結果と一致するようにユーザーに提示する WYSIWYG（What You See Is What You Get; ウィジウィグ）というユーザーインタフェースが採用されている．WYSIWYG によって，ユーザーは円などの図形や，写真，フォントを設定した文字などのデザイン要素を直感的に配置することができ，またその見たとおりのものを印刷結果として得ることができる．しかし，コンピュータ上でしかデザインを見ていない場合，いざ印刷し，掲示したときに,「コンピュータ上で見ていると気づかなかったが，周囲の物体の影響で想定していた以上に目立たない」「文字が思ったよりも小さく，見づらい」といった問題が発生しうる．そのため，制作物が，実際の環境の中でどう見えるかを確認するためには，試し刷りを行い，実際の環境に掲示する必要がある．あるいは，掲示する環境の写真を撮影し，制作物を合成することで，環境中に置かれた時にどう見えるかをシミュレーションする場合もある．しかし，これらの操作は煩雑である．また実環境で直接デザイン作業をすることは難しいため，実環境を体感しながら，その場でのアイデアをすぐにデザインに反映することはできない．さらに，デザイナーが直接現地に赴けない場合も存在する．この場合，第三者に確認を行ってもらう必要があるが，第三者がデザインの専門家でない場合は，DTP ソフトウェアを扱うことができず，デザイン上の意見について意思疎通が難しい．

1.2　研究目的とアプローチ

前節にて挙げた，コンピュータでデザイン制作を行っても，最終的な掲示環境での見え方を確認するのは煩雑・困難であるという問題の解決を目指し，実環境でのグラフィックデザインの編集及びプレビューを可能にすることを本研究の目的とする．これにより，試し刷りの煩雑さを解消し，実環境の雰囲気を体験しながらのデザイン作業を可能とする．例えば，大学の掲示板に掲示するポスターなどを，その掲示板という実環境でプレビュー・編集することで，前節にて述べた「コンピュータ上で見ていると気づかなかったが，文字が小さい」といった問題にその場で気づくことができ，またどのフォントサイズが適切なのかを試すことができる．

実環境でのグラフィックデザインのプレビュー及び編集を可能にするため，本研究ではス

マートフォン及び拡張現実 (Augmented Reality; AR) 技術を用いる．本論文において拡張現実または AR という言葉は，カメラから得られたライブビデオに対し，ライブビデオ中のマーカーや特徴的な写真の 3 次元的な位置・姿勢を推定し，コンピュータにてレンダリングした画像や 3D オブジェクトを，幾何学的整合性を保ち重畳表示する技術を指す言葉として用いる．現実世界の物体に対し，リアルタイムで情報を付加あるいは既存の情報を変化させるという AR の手法を用い，周りの環境と制作中のグラフィックデザインを重畳表示することで，ユーザーにあたかも現実に掲示したグラフィックデザインをその場で編集・プレビューしているように体感させ，周囲の環境を考慮したデザイン作業を可能にさせると考えられる．このため，本研究における AR では情報を単に仮想物体として重畳表示する以外に，仮想物体として描画されたグラフィックデザインを「編集する」というインタラクションが生じる．

そこで，本研究ではデザインのプレビュー及び編集に特化した AR である Design Modification AR (DMAR) を提案する．DMAR において達成されるべき各タスクを提案し，そのインターフェース設計を考察する．さらに，DMAR によるデザインの編集・プレビューシステムとして，すでに印刷されたデザイン中のフォントの変更をプレビューできるプロトタイプのスマートフォンアプリケーションを開発し，予備評価を行う．最終的に，上述した一般のデザイン作業フローなどにおいても有効な，DMAR インタフェースを備えたスマートフォンアプリケーション及び，主なデザイン作業を行うコンピュータ側の DTP アプリケーションを開発する．二者はインターネットを介して連携する．これにより，コンピュータによるデザイン作業，スマートフォン及び DMAR インタフェースを用いた現場を体感したプレビュー及び編集という作業形態を可能にする．

ユーザーは，コンピュータ側で実際に元となるデザインを制作する**デザイナー**及び，制作されたデザインを実環境で試し，改良を行う**テスター**を想定する．さらに，デザイナーとテスターが同一人部であり，同時に作業しないケース（**逐次作業**）と，デザイナーとテスターは異なる人物であり，遠隔で同時に作業するケース（**同時作業**）の 2 ケースを考慮する．

1.3 研究の位置付け

本研究は，まずコンピュータによるデザイン編集の支援という観点から見ることができる．DTP ソフトウェアに代表されるように，コンピュータによるデザイン編集は昨今では当たり前の作業となっている．今までは，ある程度大型の液晶ディスプレイ及びマウスとキーボードを備えたコンピュータにおいて行われきたが，現代では，スマートフォンやタブレットの普及に伴い，小さな画面においても，タッチパネルに対して操作を行うことで，直感的・簡易的なデザイン編集や画像編集が可能となっており，商用アプリケーションも多く普及している．

また，後述するプロトタイプシステムにおける，すでに印刷されたポスターと拡張現実によりインタラクションする手法は，ライブビデオを通じたインタラクションと言うことができる．ライブビデオを通じたインタラクション手法として，ライブビデオ上に写った，離れた距離にあるディスプレイを，そのライブビデオ上にて操作する手法などが研究されている．

さらに，Cashman と Grief の提唱したコンピュータ援用協調作業 (Computer Supported Co-operative Work; CSCW)[1][2] というコンテキストにおいては，Rodden[3] らの分類における，「複数地点」かつ「同期型」の協調作業の援用と考えることができる．これは，前節にあげたデザイナーとテスターが同時に作業する場合に該当する．さらに，AR による協調作業支援の観点からは，専門家と現場作業者の遠隔協調作業に関する研究であると言える．また，モバイル端末においてタッチパネルを介した AR とのインタラクション手法の研究としても位置付ける事ができる．より詳細には，第 2 章にて，関連研究と比較して説明する．

1.4 構成

本論文の構成は以下のとおりである．本章では，DTP ソフトウェアによる印刷物のデザイン工程における問題点を示し，問題解決に向けた目的とアプローチを述べた．第 2 章では，コンピュータを用いたデザイン支援及び，本研究でのアプローチとなる AR インタラクション手法に関連した研究について述べる．第 3 章では AR インタフェースの設計方針について示し，第 4 章では，すでに掲示された既存のポスター中のフォント変更を可能にするプロトタイプシステムについて述べ，第 5 章では仮想ポスターによるデザインシステムを提案し，概要とユースケースについて述べる．第 6 章では実装方法について詳述し，第 7 章において，評価実験とその結果に対する考察を述べる．第 8 章では本研究での総括と今後の発展についてまとめる．

第2章　関連研究

本章では，前章にて説明した本研究の位置づけを踏まえ，関連研究について述べる．

2.1　ARによるデザイン編集支援

デザインのプレビューをARで行う研究は90年代から行われており，特にプロダクトデザインがその研究対象となっている．Ahlersらは，部屋に設置したビデオカメラを用いて，部屋に家具を配置したらどうなるかを，コンピュータにて確認し，協調作業ができるシステムを提案している[4]．また，Klinkerらはヘッドマウントディスプレイ(Head Mounted Display; HMD)を用いたARにて，デザインした車の3Dモデルの周りを歩きながら見ることのできるシステムを提案している[5]．Leeらの Augmented Foam[6]では，CNCにより作成したラフなモックに，手などの遮蔽を考慮した上でARにより色付けを行い，マグカップなどの製品をARによりプレビューできる．本研究では対象が印刷物であり，端末はスマートフォンである．また，プレビューのみならず修正を行う点において，これらの研究とは異なっている．

2.2　ARによる協調作業支援

ARによる協調作業支援は，VRを用いたものより約10年ほど遅れ，90年代後半から様々な用途に向けて開発が行われてきた[1]．その多くは，ARの利点を活かした対面協調型のものである．すなわち，1つのテーブルなどの対象物体の周りを複数のユーザーが囲んでいるものである．対象物体の上にはARにて3Dの物体が重畳表示され，それらの物体に対して操作を行う．Kiyokawaらはテーブル上に重畳表示される3Dオブジェクトを，HMDを身につけた複数ユーザーで囲み，協調作業するシステムを提案している[7]．

一方で，遠隔協調型のものもある．Kimらは，HMDおよびカメラを装着した現場での作業者と，遠隔地にいる専門家からなる遠隔協調型のシステムを提案している[8]．作業者のカメラからのライブビデオに対して専門家はアノテーションをつけ，作業者はHMDを通して，アノテーションとともに現実を見ることで作業を行う．

本研究での仮想ポスターによるデザインシステムでは，現場でデザインをプレビュー・編集するテスターと，コンピュータにて主となるデザインを制作するデザイナーからなるため，ARにより遠隔協調型の協同作業を支援するという点においては，これらの研究と同様である．一方で，本研究ではテスターとデザイナーが同一人物であるケースがあり，デザイナーとテスターは必ずしも同時刻に作業しなくてよいため，これらの点において異なっている．

2.3 ビデオを通じた現実とのインタラクション

本研究のプロトタイプシステムでは，すでに印刷され現実の環境に掲示されたポスターと，ライブビデオを通じてインタラクションし，ポスター中のフォントの書き換えを可能とする．ライブビデオを通して現実の物体とインタラクションを行う研究として，Rohs は 2 つのマーカーを用い，ユーザーが画面内の座標系でタップした位置と，現実の座標系を変換することで，現実の地図や表中の情報を選択する手法を提案している [9]．本研究のプロトタイプシステムはマーカーレス AR であり，ユーザーが選択した領域はトラッキングによって保持する．また，Schoning らは，現実の地図上に施設のアイコンをオーバーレイして表示し，タッチペンによる画面の 2 点のタップにより，地図上での距離を表示するなどのインタラクションを行う手法を提案している [10]．本研究ではオーバーレイのみならず，既存の情報を修正し，インタラクションはタッチジェスチャによって行う．

2.4 モバイル端末上での AR インタラクション

スマートフォンのようなハンドヘルド型デバイスのカメラおよび液晶ディスプレイによるビデオシースルーによる AR におけるインタラクションには，大きく分けて 2 つのインタラクション手法が存在する．ひとつは，AR 環境として表示される，カメラに写る範囲に手や道具を差し入れてインタラクションする方法である [1]．この場合，片手でデバイスを保持し，もう一方の手によりインタラクションを行う．Wolfgang らは，カメラに写る空間に手を差し入れ，仮想物体の選択や移動を行うインタラクション手法を提案している [11]．Yousefi らは同様にカメラに写る空間に手を差し入れ，仮想物体をつまむようなインタラクションにより，仮想物体の拡大・縮小や回転を行う手法を提案を提案している [12].

もうひとつのインタラクション手法はデバイス付属のタッチスクリーンを用いたインタラクション手法であり，本研究でのインタラクション手法もこれに類する手法である．Kasahara らはタブレット端末を用い，タブレットをかざした実空間中の場所に，タッチスクリーンにより絵を描き，アノテーションを付与し共有するインタラクション手法を提案している [13]. また，仮想物体をタッチにより操作する手法も複数提案されている．Mossel らは 3 次元の仮想物体を，初回のタッチと改善の 2 ステップにより，仮想物体が入り組み，一部が隠れている場合でも正確に任意のオブジェクトを片手の 1 本指のタッチジェスチャにて選択できる手法を提案している [14]．さらに，タッチジェスチャのみならず端末の回転を組み合わせ，仮想物体を，実空間から来る配置の制約（i.e. マーカーによって決定される 3 次元空間）から一時的に切り離す機能を加える事により，複雑な回転や配置を可能にした手法も提案している [15]．　本研究のシステムでは，実空間中の既存のポスターを対象とするプロトタイプシステムと仮想ポスターを操作対象とする提案システムのどちらにおいても，タッチスクリーン上のジェスチャにより操作を行う．その対象は仮想ポスターを構成する，矩形やテキスト，写真などのデザイン要素である．上記の手法 [14][15] では操作対象の仮想物体は 3 次元であるが，本研究では操作対象は 2 次元のポスター上のデザイン要素であり，移動や拡大や縮小といった変更を加えた結果も，ポスター平面上の 2 次元内に制限される．そのため，2 次元の

ジェスチャであるタッチスクリーン上のジェスチャを出来る限り活用することで，使いやすいインタフェースを目指した．このインタフェースの設計については第 3 章および第 5 章にて詳述する．

第3章　DMARインタフェース

本章では，制作されたデザインを，ユーザーが実環境にてプレビュー・編集を行うと想定したときに，行われるであろうタスクについて考察し，そのタスクをスマートフォン上でARにより行う場合に各タスクはどのようなものとなるかを考察する．

まず，本研究ではスマートフォンをDesign Modification Augmented Reality (DMAR) インタフェースに用いるデバイスとして採用する．理由として，スマートフォンは現在最も普及しているモバイル端末であり，多くのユーザーが慣れている点，通常の業務にも導入しやすい点が挙げられる．また，タッチスクリーンを介し，ARによって描画された仮想物体との直感的なインタラクションが可能である．ユーザーの視界の上にそのまま重畳できるという利点から，ヘッドマウントディスプレイ (Head Mounted Display; HMD) やグラス型端末の採用も考えたが，ヘッドマウントディスプレイはその重量及び，ユーザーからは外が全く見えない点から，現在のグラス型端末では表示領域が極めて小さい点から，採用を見送った．

次に，本研究において想定する，ユーザーが実環境にてデザインを見てどう編集したらよいかを考えている様子を図3.1に示す．図中にて左にいるユーザーAは,「もしフォントを変えたら，より見やすいかもしれない」と考え，右にいるユーザーBは「背景色をピンクに変えたら周りの環境から目立つかもしれない」と考えている．ユーザーはこのように様々な改善案を，様々な角度から見たり，距離をおいてみたり，と見え方を変えながら想像すると考えられる．この想像を確認できるようなシステムが存在したとして，そのシステムでユーザーが達成する必要がある作業について考えると，まずは，その実環境にポスターなどのデザインを配置する作業が発生するだろう．さらに，図中のユーザーの想像している案をプレビュー・編集できるようにするためには,「どの部分」を「どのように」変更するかを，何らかの形で表現する必要がある．図中の例で言うと「一番上のキャッチコピーの見出しの部分」の「フォントを変更する」,「塗りつぶされている背景」の「色を変更する」ということである．また，ユーザーはその結果をプレビューできる必要がある．この場合で言うと，フォントを変えた結果や背景色をピンクに変えた結果をその場で確認できることである．

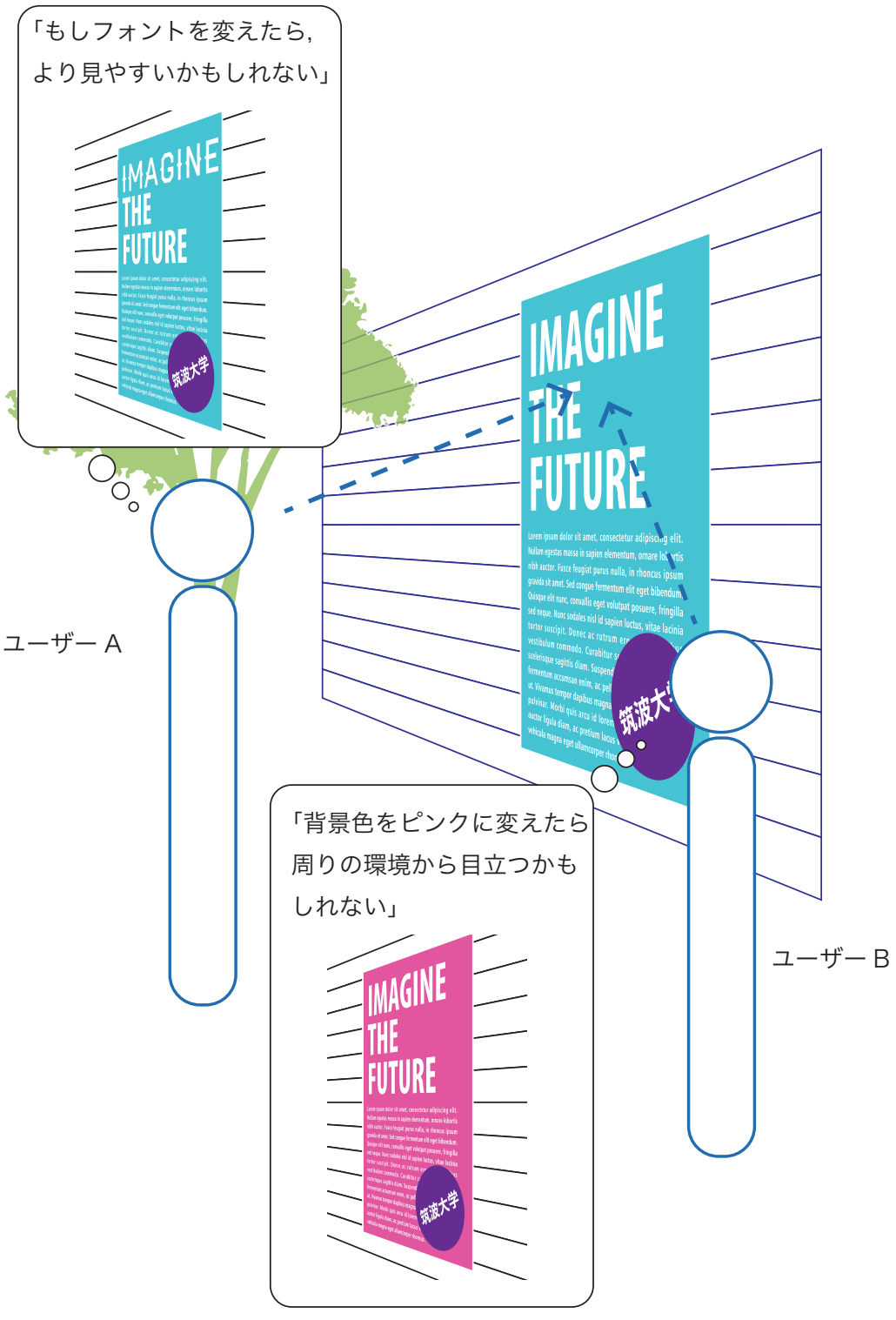

図 3.1: 実環境でデザインを確認する様子

3.1 スマートフォンを用いたARによるデザインのプレビュー及び編集

以上のような想定環境において，スマートフォンを用いて，実際の環境においてデザインのプレビューおよび編集を拡張現実により行う想定シーンを図3.2に示す．以降では簡単のために，掲示する印刷物をポスターと総称するが，フライヤーや料理店のメニュー，より大きい広告などでも同様である．ユーザーは，編集中のポスターを掲示する壁などにスマートフォンをかざし，ポスターをARにより「貼り付ける」ことで，あたかもその壁に掲示されているポスターを直接編集しているかのようにデザイン作業を行う．あるいは，すでに掲示されている現実のポスターにスマートフォンをかざし，ARにて書き換える．図3.2中上図は，ユーザーがDMARによりプレビュー及び編集している様子を表している．ここでは，ユーザーがポスターを貼り付けた領域を薄桃色で示している．下図は，ARによりポスターが描画されたスマートフォンの画面である．下左図は，ポスターを貼り付けて正面から見ている様子，下右図では，DMARインタフェースにより背景色を編集し，変更した様子を表している．

このように，実環境においてプレビュー・編集を可能にするARシステムについて考察したとき，DMARインタフェースにおいて行われるタスクとして，大別して4種類のタスクがあると考えられる．これらをDMARインタフェースにおける4タスクとして提案する．

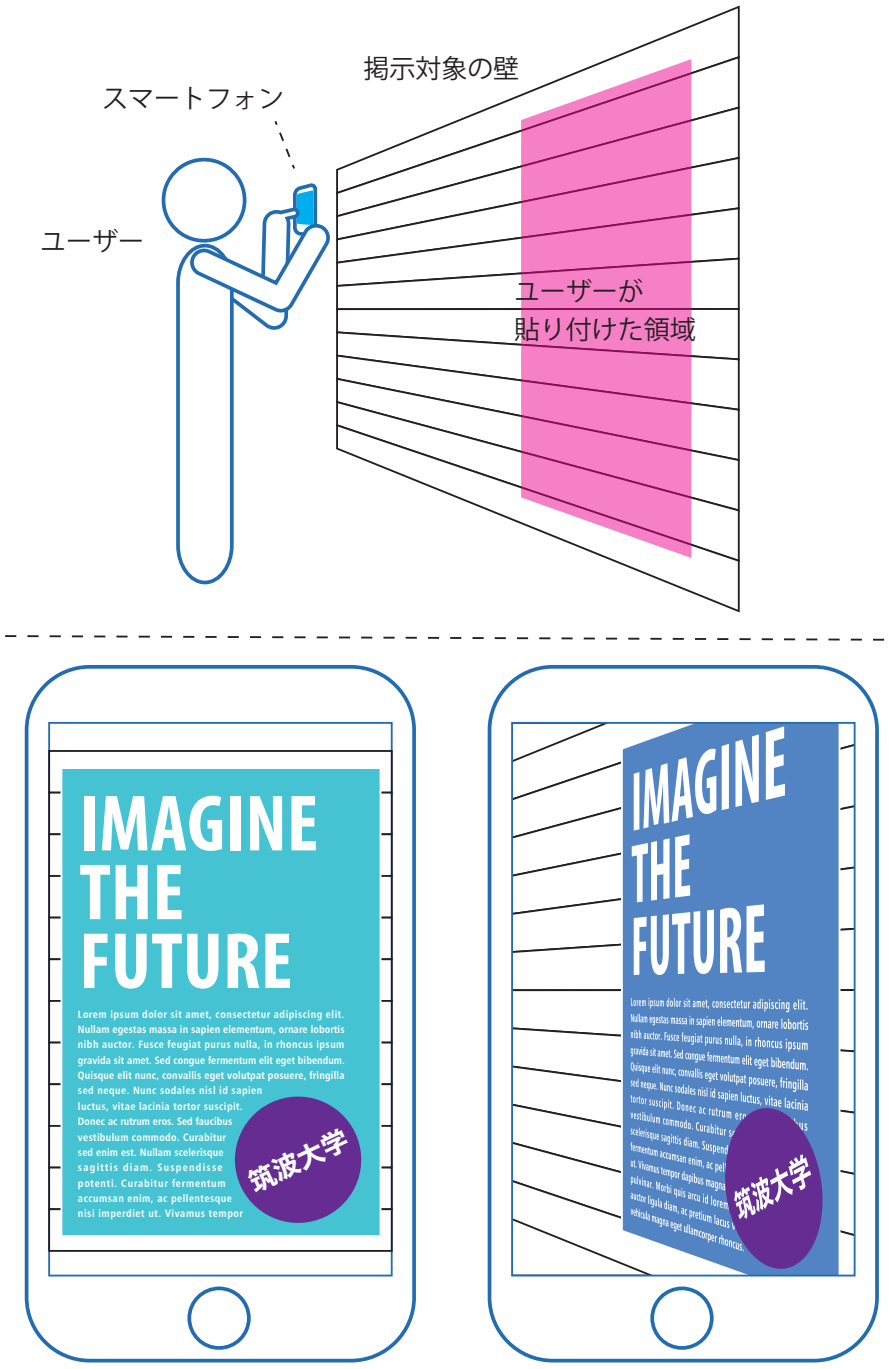

図 3.2: 上：DMAR によるプレビューおよび編集中の様子；下左：ポスターを AR により貼り付け，正面から見た様子；下右：背景色を編集し，別の角度から見た様子

3.2 DMARにて行われる4タスク

各タスクについて詳述する前に，DMARで行うタスクの対象を明確にする．本研究では，ポスター内の要素を**グラフィック**と呼称することにする．すなわち，DTPソフトウェア上でユーザーが作成し移動・拡大，色の変更などを行える，デザインを構成する各要素を指す．以下にグラフィックの例を挙げる．

- 矩形
- 楕円形
- テキスト
- 図形

前述した想定環境でのユーザーの想像と照らし合わせると，見出しのフォントの変更は，テキストグラフィックを対象とし，フォントを変更する，という編集を行うことになる．また，背景色を変更する例では，背景領域の矩形グラフィックを対象とし，色を変更するという編集を行うということである．そこで，これらの編集を表す4つのタスクを提案する．

1. ポスターの掲示位置の指定
2. 編集するグラフィックの選択
3. 編集内容の入力
4. 編集結果の確認

以下に，各タスクについて詳述する．

1. ポスターの掲示位置の指定

ARを用いて，プレビュー及び編集するポスターを，ARに用いてどこに貼り付けるかを指定する操作である．インタラクションメソッドとしては，ドラッグによる選択や，すでに実環境に存在しているポスターをタップによる指定する，などが考えられる．

2. 編集するグラフィックの選択

ポスター中のどのグラフィックを編集するかを指定する操作である．色を変更したい図形や移動したいテキストなどをタップすることで選択する，といったインタラクションメソッドが考えられる．

3. 編集内容の入力

選択したグラフィックに変更を加える．図形であれば色や大きさ，位置などを変更する．例えば背景色の水色を赤色に変える，などである．これは，編集対象のオブジェクトによってインタフェースを変える必要がある．例えば，テキストのときはフォントの変更を可能にする必要がある．

4. 編集結果の確認

ユーザーが選択したグラフィックに対して加えた編集内容を反映させ，ユーザーに提示する．WYSIWYG のようにわかりやすいインタフェースを実現するためには，ユーザーの入力をリアルタイムで反映させ描画する必要がある．また，ユーザーに，実環境を体感しながら，異なる角度から見たらどうか，遠ざかってみてみた時の印象はどうなるか，などを試せるようにするため，AR にて 3 次元空間に描画する必要がある．

これらのタスクに対する，プロトタイプシステムでのインタラクションメソッドについて次章で述べ，プロトタイプシステム開発から得られた知見を元にした最終的な提案システムについては 5 章以降にて述べる．

第4章　既存のポスターを編集するプロトタイプシステム

本章では，すでに印刷され，現実環境に掲示されたポスター中のフォントを編集・プレビューするプロトタイプシステムについて，DMAR インタフェースの設計および実装と，本システムを用いてデザインの修正作業が行えるかを調査するための予備実験と結果に対する考察を述べる．

4.1　システム概要

本システムでは，印刷済みのポスター中のグラフィック要素をリアルタイムに編集し，結果をプレビューできる．すでに印刷済みのポスターを画像処理により編集することで，掲示された環境の照明色などが反映された状態のポスターを元にした実環境でのリアルタイムな編集を可能とする．可能な操作として，ポスター中のテキストのフォントの変更を対象とする．理由は，文字は印刷物中に最も頻出するデザイン要素だからである．すなわち，色やサイズを保ったまま，文字列のフォントの変更を行える DMAR インタフェースを設計し，実装を行う．本システムの概要を図 4.1 に示す．図に示すように，ユーザーは修正したいポスターにスマートフォンをかざし，アプリケーションを操作することで，図中のスマートフォンの画面のように，フォントを変更したプレビューを見ることができる．

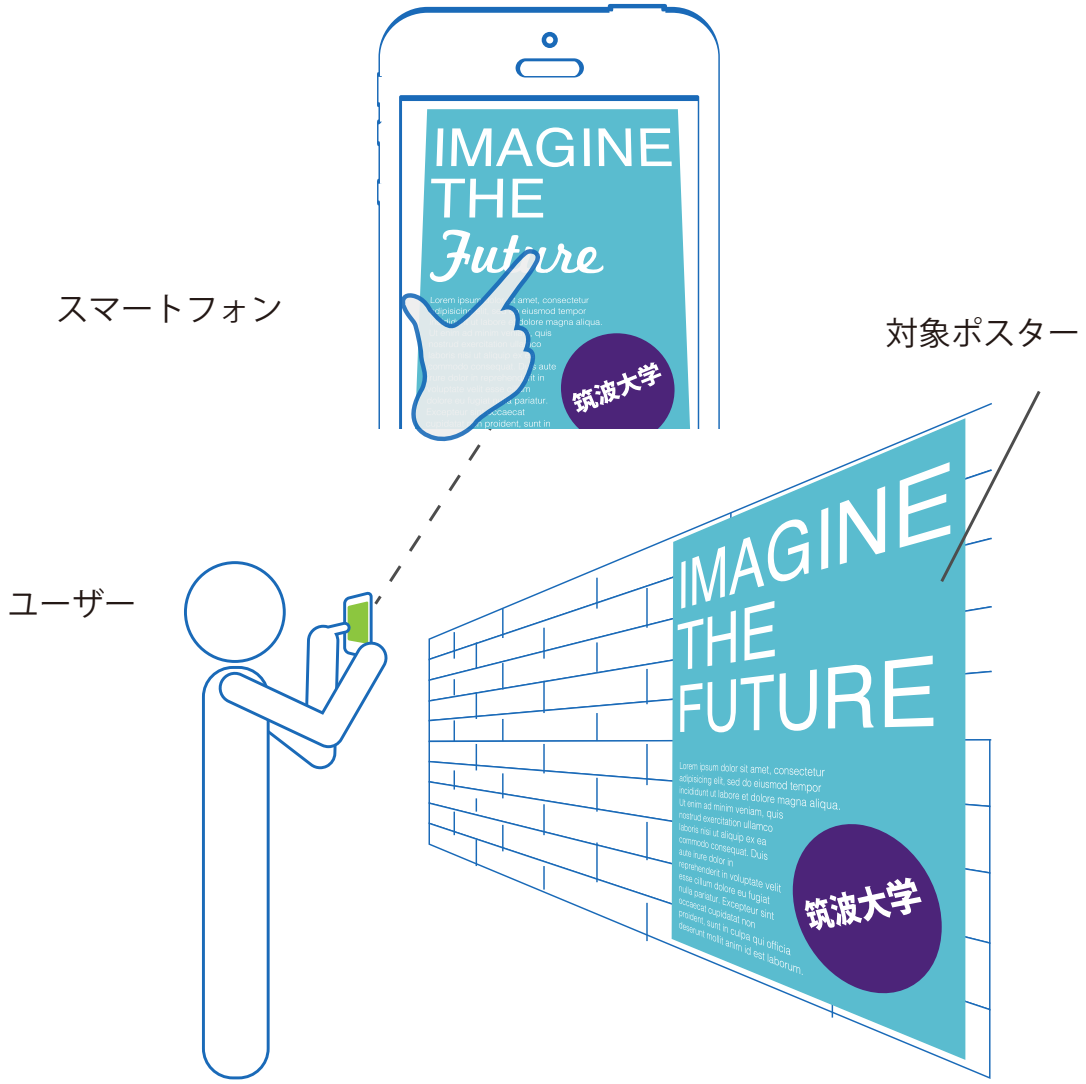

図 4.1: 既存のポスター編集システムにてフォントを変更する様子

4.2 DMAR インタフェースのインタラクションメソッド

本節では，前章にて定義した DMAR インタフェースでの各タスクに対するプロトタイプでのインタラクションメソッドについて述べる．

対象ポスターの指定

対象ポスターの指定は，現実に印刷されたポスターが存在するため，単にカメラにポスターをかざすだけである．ユーザーは修正領域の選択から操作を開始することができる．

編集するグラフィックの選択

ライブビデオ中の修正したい文章領域をドラッグする操作で選択させる．ユーザーのドラッグ中に，半透明のラバーバンドを表示することで，ユーザーに選択中の領域をフィードバックする（図 4.2）．指を離すことで修正領域は確定し，即座に選択した領域中の文字のフォントが置き換わる．

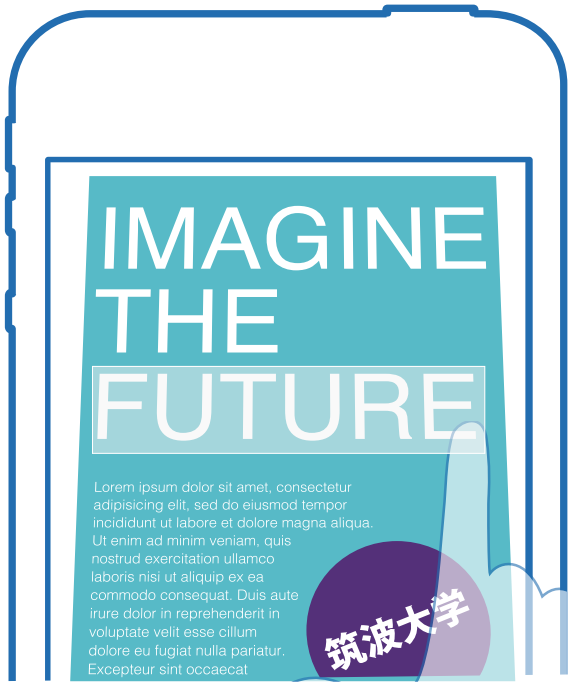

図 4.2: ドラッグによる文字領域の選択

編集内容の入力

ユーザーは修正領域をスワイプすることにより，数あるフォント見本からひとつひとつを取り出すかのように，異なるフォントでの見え方を確認することができる．それぞれのフォントは，すでに修正領域中の文章で描画されているため，様々なフォントでの見え方を素早く確認することが可能である（図 4.3）．

図 4.3: スワイプによるフォントの変更

編集結果のフィードバック

元の文章を新しいフォント及び修正領域から抽出した文字色にてレンダリングしたものが修正領域にオーバーレイされる．元の修正領域の背景の復元により，元々の文字領域は消去され，あたかも，元の修正領域を書き換えたかのように，修正結果を確認することが可能である．これらの処理はリアルタイムで行われ，ユーザーが端末を動かしても，この修正のプレビューは追従する．

4.3 実装

本システムをスマートフォン (iPhone 6, 4.7inch, 1334x750, 326dpi) 上で動作するアプリケーションとして実装した．開発には Mac OS X 10.10 の Xcode 6.1, iOS SDK 8 で開発し，画像処理には OpenCV を用いた．実装したアプリケーションを用いた操作の流れを，図 4.4，4.5，4.6 に示す．それぞれ，異なったポスター 2 点に対して操作を行っている様子を撮影した．

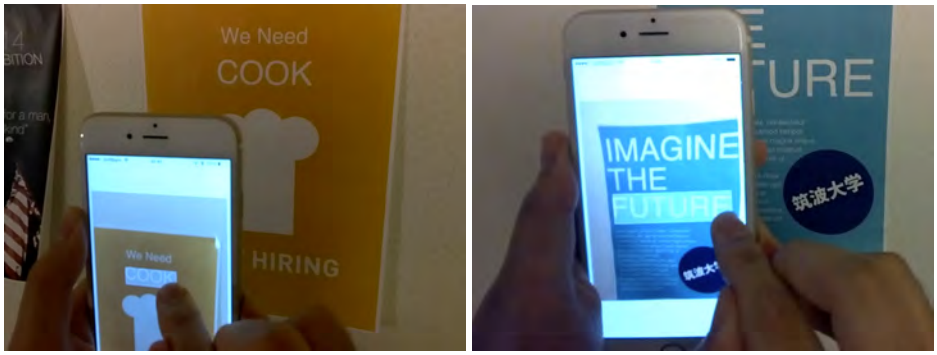

図 4.4: ドラッグによる修正領域の選択

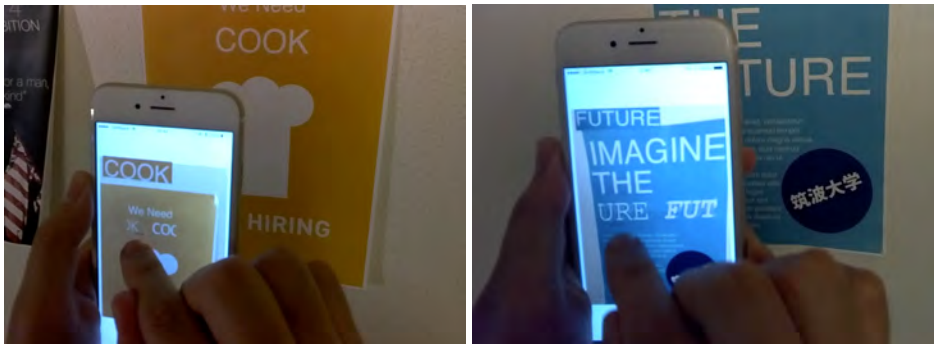

図 4.5: スワイプによるフォントの切り替え

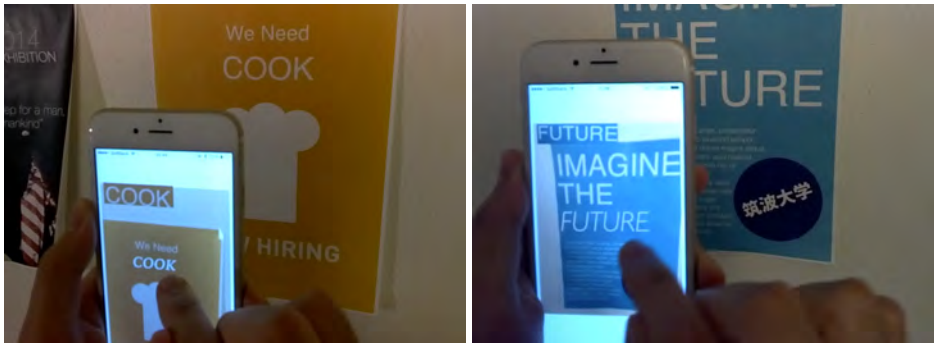

図 4.6: フォントを修正し終えた状態

それぞれの左図ではポスター中の “COOK” という単語を，右図では “FUTURE” という単語を修正している．図 4.4 ではドラッグによる修正領域の選択を，図 4.5 では修正内容の入力

（スワイプによるフォントの切り替え）を，図 4.6 では修正結果のフィードバック（スワイプ動作を終えた状態）を示している．

全体の処理の流れを以下に示し，各処理について詳述する．

1. カメラ入力の取得
2. ユーザーにより指定されたテキスト領域の取得
3. テキスト領域中の文字の認識
4. テキストの色の抽出
5. テキストの文字領域削除による背景画像の生成
6. テキスト領域のトラッキング
7. 背景画像に様々なフォントにてテキストをオーバーレイ

1. カメラ入力の取得

iPhone に搭載されているカメラからリアルタイムのライブビデオを取得する．実装には iOS SDK に含まれる，動画処理フレームワークである AVFoundation を用いた．

2. ユーザーにより指定されたテキスト領域の取得

ユーザーはタッチスクリーン上でドラッグ操作をすることにより変更したいテキスト領域を選択する．取得したカメラからの画像中の，ドラッグの開始点及び終了点から決まる線分を対角線とする長方形をテキスト領域とする．ユーザーが，図 4.4 左のように選択した場合のテキスト領域を図 4.7 に示す．この場合，ユーザーはポスターに対してスマートフォンを正面からかざしているため，元のポスターはおおよそ長方形となるため，ユーザーが選択した領域中のテキストも歪んでいないことを期待できる．

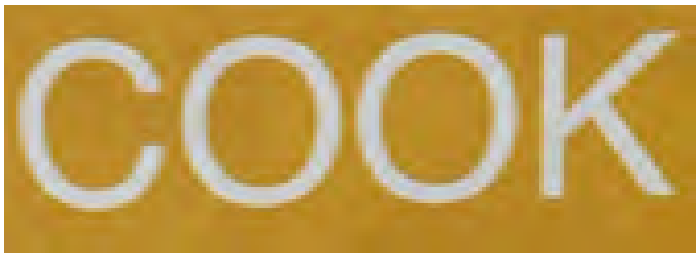

図 4.7: 選択されたテキスト領域

3. テキスト領域中の文字の認識

テキスト領域を適切な閾値により二値化し，文字認識エンジンである Tesseract[1] に入

[1]Tesseract,
https://code.google.com/p/tesseract-ocr/(2014.11.24),
Tesseract-OCR-iOS, https://github.com/gali8/Tesseract-OCR-iOS

力する．二値化したテキスト領域を図 4.8 に示す．

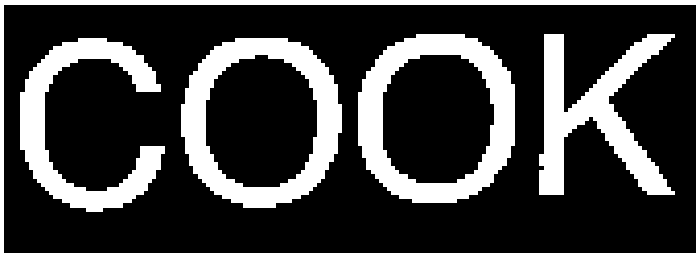

図 4.8: 二値化したテキスト領域

4. テキストの色の抽出

テキストの色を保持したたまのフォントのプレビューを可能とするために，指定されたテキストの色を抽出する．まず，テキスト領域中から，文字領域のみを抽出する．これには，図 4.8 に示した二値化画像をマスクとして使う．次に，マスクされた画像内で最も頻出する色を求め，そのテキストの文字色とする．しばしば二値化画像は白黒が反転している場合があるため，誤って背景色を文字色と認識してしまうケースがある．その対策として，テキスト領域の境界付近に白い画素が多く中央付近に黒い画素が多い場合には，反転した後に前述の最頻出色の計算を行っている．

5. テキストの文字領域削除による背景画像の生成

認識したテキスト及び色にて新しいテキスト画像を生成する前に，テキスト領域の背景を復元する必要がある．そのために，Inpaint アルゴリズムを用いている．Image Inpating[16] アルゴリズムは傷ついた写真などの傷領域を，その周辺領域を元に復元するアルゴリズムであり，OpenCV にも実装されている．本システムでは，テキスト領域を傷領域として Inpaint アルゴリズムに入力することで，テキストを削除し背景を復元し背景画像を生成する．図 4.9 に生成した背景画像を示す．図ではユーザーが選択した “COOK” という文字が消え，背景色が復元されている．

図 4.9: 選択された領域中のテキストが削除され，背景色が復元されたポスター

6. テキスト領域のトラッキング

フォントのプレビューを，カメラからの入力画像にリアルタイムにオーバーレイするためには，選択されたテキスト領域のトラッキングを行う必要がある．本システムでは，トラッキングアルゴリズムとして Clustering of Static-Adaptive Correspondences for Deformable Object Tracking (CMT) [17] アルゴリズムを用いている．CMT は拡大縮小と回転に頑健なトラッキングアルゴリズムである．これにより，ユーザーがスマートフォンを回転させたり，ポスターから遠ざかったり近づいた時にでもフォントのプレビューを追従させオーバーレイできる．

7. 背景画像に様々なフォントにてテキストをオーバーレイ

最後に，カメラからの入力画像に対して，背景画像及びフォントのプレビューをオーバーレイして表示する．CMT によるトラッキング結果から，サイズ及び角度を再計算して，最初にユーザーが選択した領域に重なるようにオーバーレイする．ユーザーはフォントのプレビューをスワイプすることで様々なフォント候補を切り替えてプレビューできる．フォント候補のスワイプには iOS SDK に含まれる，スクロールを可能にするビューである `UIScrollView` を用いた．図 4.10 に，最終的なフォントのプレビューの様子を示す．この図では，元のフォントである Helvetica が TimesNewRoman に，元の

図 4.10: 選択されたテキストのフォント変更プレビュー

色と大きさを保ちながら変更されている．

4.4 予備実験

4.4.1 実験内容

本システムにて実際にデザインの修正作業が行えるか，インタフェースに問題点はないかどうかを調査するために予備実験を行った．被験者は，デザインを専攻している大学院生 3 人（23〜27 歳，男）である．スマートフォン上に実装した本システムを，図 4.11 および被験者の周囲に合ったポスターや本の表紙などに対して使用させ，本システム及びその DMAR インタフェースについてインタビューによりコメントを得た．

4.4.2 実験結果

本システムの有用性については,「実際の公共空間でサインやポスターの誘目性を調べられるのは良い」「自分のではないポスターに対して適用すると，元々良いデザインを変えたらどうなるかをすぐに試せるため，デザインの学習に役立つ」との回答であった．また，インタフェースについては「周囲の環境を含めて，直感的な操作で視覚的印象をプレビューできる」と回答を得た．

図 4.11: 予備実験に用いたポスター

4.5 考察とまとめ

予備実験により，システムの有用性について確認を行った．現場での，すでに実環境において掲示されているポスターのリアルタイムでの編集は，デザイン作業において有用だとの回答を得ることができた．また本システムの DMAR インタフェースの操作性についても直感的との評価を得，AR によるデザインのプレビューおよび編集というタスクを効果的に行えることを確認した．

以上のプロトタイプシステムによる知見を活かし，さらに複雑なデザイン作業に対応できるシステムの開発を行う．対象とするのは，主となるデザインの制作と，現場での確認，それを踏まえての再修正などのいくつかのプロセスからなるデザイン作業である．このようなデザイン作業においても対応できるシステムを開発することにより，ユーザーはコンピュータ上で元となるデザインを制作し，そのデザインを実環境にて，印刷の手間なしに確認し，その結果を元に，さらにコンピュータ上で作業を継続することができる．

第5章　仮想ポスターによるデザインシステム

5.1　システム概要

本システムは，前章にて説明した既存のポスターを編集するシステムを元に，さらに複雑なデザイン作業に対応できるよう開発を行ったものである．プロトタイプシステムと同様に，実環境におけるデザインのプレビュー・編集を可能にするシステムである．本システムでは，主となるデザインの制作と，現場でのプレビュー・編集，それを踏まえての再修正を行うことができる．

プロトタイプシステムはすでに実環境に存在するポスターを元にプレビュー・編集を行うシステムであったが，本システムは，AR により仮想ポスターを，あたかも実環境に掲示したかのようにプレビュー・編集できるシステムである．制作したデザインを仮想ポスターとして描画することにより，ユーザーは制作したデザインを，印刷の手間なしに実環境にてプレビュー・編集することが可能である．またデザインの元データを用いることにより，プロトタイプシステムでは不可能だった，各デザイン要素の移動，色の修正などを可能とする．AR による仮想物体の描画では，しばしば現実の物体と仮想物体の色合いが一致しないなどの光学的整合性の問題が発生し，デザインのプレビューにおいては大きな問題となるが，本システムでは第 5.5.2 節にて述べる白色領域指定による参考色乗算法という手法を考案し，照明色の仮想ポスターへの反映を可能にした．

システムは，主となるデザインを制作する Mac OS X 用 DTP アプリケーション（**Base**），DMAR インタフェースにてデザインのプレビュー･編集を行う iOS 用アプリケーション (**Client**) 及び，二者間を中継するサーバー（**Relay**）によって構成されている（図 5.1）．本章では，まず，前章にて定義した DMAR インタフェースについての本システムでの設計を述べる．次に，1 枚のポスターなどを表す構造である**ドキュメント**及びそれを構成する**グラフィック**について説明した後，ユーザーから見える Mac OS X 用アプリケーション及び iOS 用アプリケーションのインタフェース及び機能について述べる．サーバー側の詳細については，第 6 章にて実装とともに述べる．

5.2　DMAR インタフェースのインタラクションメソッド

本節では，第 3 章にて定義した DMAR インタフェースの各操作に対して，本システムにおけるインタラクションメソッドについて説明する．タッチパネル及び AR の特性を活かし，直感的なインタフェースを目指す．前述のとおり，本システムでは DMAR はテスターによる使

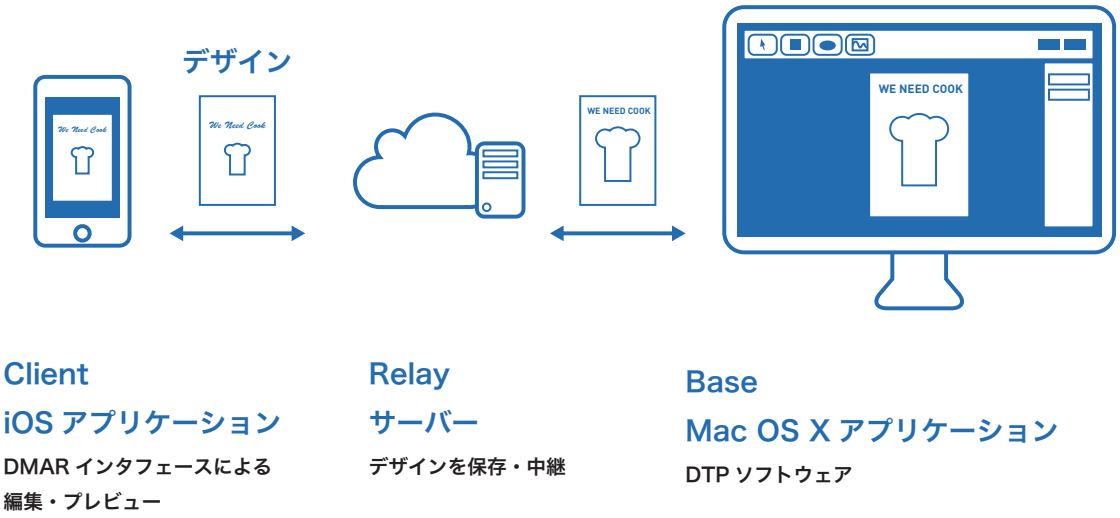

図 5.1: システム全体図

用を想定しており，必ずしも DTP ソフトウェアの扱いに長けているユーザーを対象とはしていない．したがって，コンピュータ側の DTP ソフトウェアが可能なすべての操作を搭載しインタフェースが複雑化することは避ける．

5.2.1 ポスターの掲示位置の指定

ポスターの掲示位置の指定に対し，「貼り付け」インタラクションを考案した．図 5.2 にインタフェースを示す．カメラから得られるライブビデオに対し，半透明にてポスター画像をオーバーレイし，ユーザーは気に入ったところで「貼り付け」ボタンを押すことで，ちょうどオーバーレイされていたところに存在していた壁や，現実のポスターなどに貼り付けることが可能となる．この動作をユーザーに行わせることで，デバイス内に存在する仮想的ポスターを，実空間に「転写」することを体感させるのが狙いである．

5.2.2 編集するグラフィックの選択

編集するグラフィックの選択は，グラフィックをタップすることにより行う．そのまま他のグラフィックをタップすることにより複数選択が可能である．個々のオブジェクトを再度タップすることにより選択解除が可能である．また，画面中のポスター以外の領域をタップすると，全てのオブジェクトを選択解除する．選択されたオブジェクトはバウンディングボックスに囲まれて表示される（図 5.3 中の黄色い矩形）．

図 5.2: 「貼り付け」インタフェース

5.2.3 編集内容の入力

編集内容の入力は，どのグラフィックでも可能な操作と，そうではない操作がある．例えば，フォントの変更操作は，単なる矩形に対しては不可能である．また，写真の色を変えることもできない．以下に，全グラフィック共通の操作と，そうでない操作を挙げる．

- 全グラフィック共通の操作
  - 移動
  - 拡大・縮小
- 図形・テキストに共通の操作
  - 色の変更
- テキスト固有の操作
  - フォントの変更

全オブジェクト共通の操作は移動と拡大・縮小である．タッチパネルの利点を活かし，移動はドラッグ，拡大縮小はピンチジェスチャによって行う．

選択対象によって可能な操作が異なるインタフェースでは，Windows や Mac OS X といった OS のように，一般的にコンテキストメニューが用いられる．したがって，本研究でもコンテキストメニューを採用した．例えば，色を変更可能なグラフィックを選択すると，そのグラフィックの近くにパレットが表示される．図 5.3 にコンテキストメニューが表示されているインタフェースを示す．図中左は，テキスト選択時であり，色変更のためのパレットとフォント変更のためのフォント候補が表示されている．パレットは横軸に彩度，縦軸に彩度をとってあり，Adobe Illustrator などにおいても用いられているものである．図中右は，楕円形選択時であり，パレットのみ表示される．また，パレット中の任意の位置をタップすることにより色を指定する．フォント一覧であれば，反映したいフォントをタップする．コンテキストメニューは，ポスター同様に AR にて 3 次元空間中に描画する．このため，ディスプレイのサイズに左右されない作業空間を確保できる．

図 5.3: コンテキストメニュー（左：テキスト選択時；右：楕円形選択時）

5.2.4 編集結果のフィードバック

編集結果のフィードバックはリアルタイムに反映されるようにする．すなわち，移動や拡大・縮小であればユーザーが指を動かすたびに位置や大きさが変わり，色の変更ではユーザーがパレット上をなぞると，選択したグラフィックはユーザーの指の下の色となる．

5.3 ドキュメント

本システムでは，ポスターやチラシなどの 1 枚のグラフィックデザインを表す構造を**ドキュメント**と呼称する．ドキュメントは ID，名前，横幅，縦幅，複数の**グラフィック**からなる．表 5.1 に構成要素をまとめる．

表 5.1: ドキュメントの構成要素

要素名	説明	例
ID	ユニークな ID	5F16330E-879B-4B3B-⋯
名前	ドキュメントの名前	“大学のポスター”
横幅	ドキュメントの横幅 [mm]	210
縦幅	ドキュメントの縦幅 [mm]	297
複数のグラフィック	ドキュメントを構成する図形や画像	長方形

本システムにおいて，グラフィックとはドキュメント内に配置する図形や画像を指す．グラフィックには矩形，楕円形，テキスト，画像の 4 種類を用意している．各グラフィックは，複数の属性を持つ．属性には，全てのグラフィックが持つ属性とグラフィック固有の属性の 2 種類が存在する．全てのグラフィックは属性として範囲を持ち，そのグラフィックが位置する座標 (x, y) 及び縦幅と横幅を規定する．さらに，矩形グラフィック及び楕円形グラフィックには属性として塗りと線，線幅を持つ．塗りと線はそれぞれ図形の内側を塗りつぶす色と，境界線を描く色を表し，線幅はその境界線の幅を表す．テキストグラフィックであれば塗り，文字列，フォントを，画像グラフィックであれば画像データを持つ．

Base, Relay, Client ではこのドキュメント及びグラフィックを操作・編集することにより，グラフィックデザインを作成する．

5.4 Mac OS X 用アプリケーション

Mac OS X 用アプリケーション (Base) は，通常の DTP ソフトウェアに似たアプリケーションである．Adobe Illustrator 及び，Apple Keynote を参考にしたインタフェースである．Apple Keynote は Mac OS X 純正の UI で，かつ，オブジェクトの配置など同様の機能を有するため，参考とした．図 5.4 に Base の画面構成を示す．図中の数字が表す部分の名称を表 5.2 に示す．

表 5.2: Base の各インタフェースの名称

数字	名称
(1)	ツールバー
(2)	キャンバス
(3)	バウンディングボックス
(4)	インスペクタ

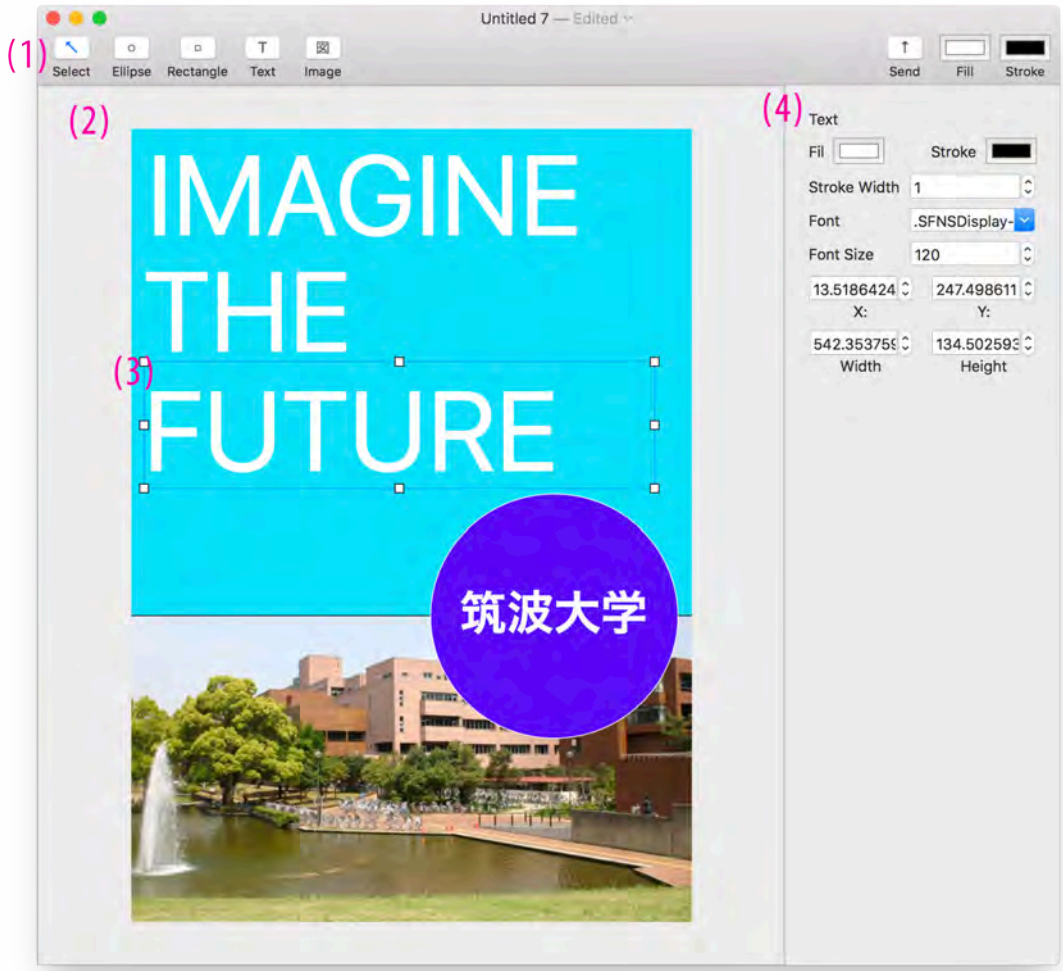

図 5.4: Base の画面構成

5.4.1 ツールバー

図 5.4 の (1) は，ツールバーである．ツールバー中の 8 つのボタンは，左から選択ツール，楕円ツール，矩形ツール，テキストツール，画像ツール，送信ツール，塗りツール，線ツールである．各ツールの機能を以下に示す．選択，楕円，矩形，テキストツールのボタンは，クリックするとハイライト表示となり，そのツールが選択されていることを表す．選択されたツールは，キャンバスへの操作に用いられる．

選択ツール

画面中のグラフィックを，クリックにより選択するツール．Shift キーを押下しながらクリックする，あるいは選択範囲をドラッグにより指定することで複数オブジェクトの選択が可能である．選択したグラフィックは，図 5.4 中 (3) のようにバウンディングボックスで囲まれて表示される．さらに，グラフィックをドラッグすることによりオブジェクトを移動できる．バウンディングボックス上の矩形（ハンドル）をドラッグすると，その方向へ拡大・縮小する．Shift キーを押下しながらドラッグすることにより，縦横比を保持したままの拡大・縮小が可能である．

また，テキストグラフィックをダブルクリックすることにより，テキストの編集を開始する．テキスト以外の部分のクリックにより，編集は終了する．

楕円ツール

楕円を描画するツールである．ドラッグすることにより，ドラッグの始点と終点を結んだ線分を対角線とする矩形に内接する楕円を描画できる．楕円は塗りツールで指定した色で塗りつぶされ，境界線は線ツールで指定した色となる．

矩形ツール

矩形を描画するツールである．ドラッグすることにより，ドラッグの始点と終点を結んだ線分を対角線とする矩形を描画できる．楕円と同様に，塗り及び線が適用される．

テキストツール

テキストを描画するツールである．ドラッグすることにより，ドラッグの始点と終点を結んだ線分を対角線とするテキスト領域を描画できる．自動で “テキスト” という文字列が代入される．前述のように，選択ツールでダブルクリックすることにより，文字列の編集が可能である．テキストには塗りツールの色のみが適用される．

画像ツール

画像をキャンバスに挿入するツールである．画像ツールをクリックすると，ダイアログが表示され，ユーザーは自身のコンピュータに保存されている任意の画像を選択することが可能である．ユーザーが選択された画像は，キャンバスの中央に追加される．

送信ツール

送信ツールボタンをクリックすることで，現在のドキュメントを Relay サーバに送信

することができる．送信中は「送信中」の表示と，アクティビティインジケータが表示され，ユーザーに待機を促す（図 5.5．アクティビティインジケータが送信が終わると，元の画面に戻る．送信自体は，通常の Wi-Fi 環境でインターネットに接続されている場合，1〜2 秒で完了する．すでに同じ ID のドキュメントが存在する場合は更新され，存在しない場合は新規作成される．

塗りツール

描画するグラフィックの塗りの色を設定する．ボタンには現在塗りとして指定している色が表示され，クリックすることで，パレットが表示される．パレット内の色を選択することで，ボタンにも色が反映される．図 5.6 に表示されるパレットを示す．

線ツール

描画するグラフィックの線の色を設定する．その他は塗りツールと同様である．


clouds_poster
Select
Ellipse
Rectangle
Text
Image
Send
Fill
Stroke
No Selection
SECURE
IN
Sending...
THE
CLOUDS

図 5.5: 送信中画面

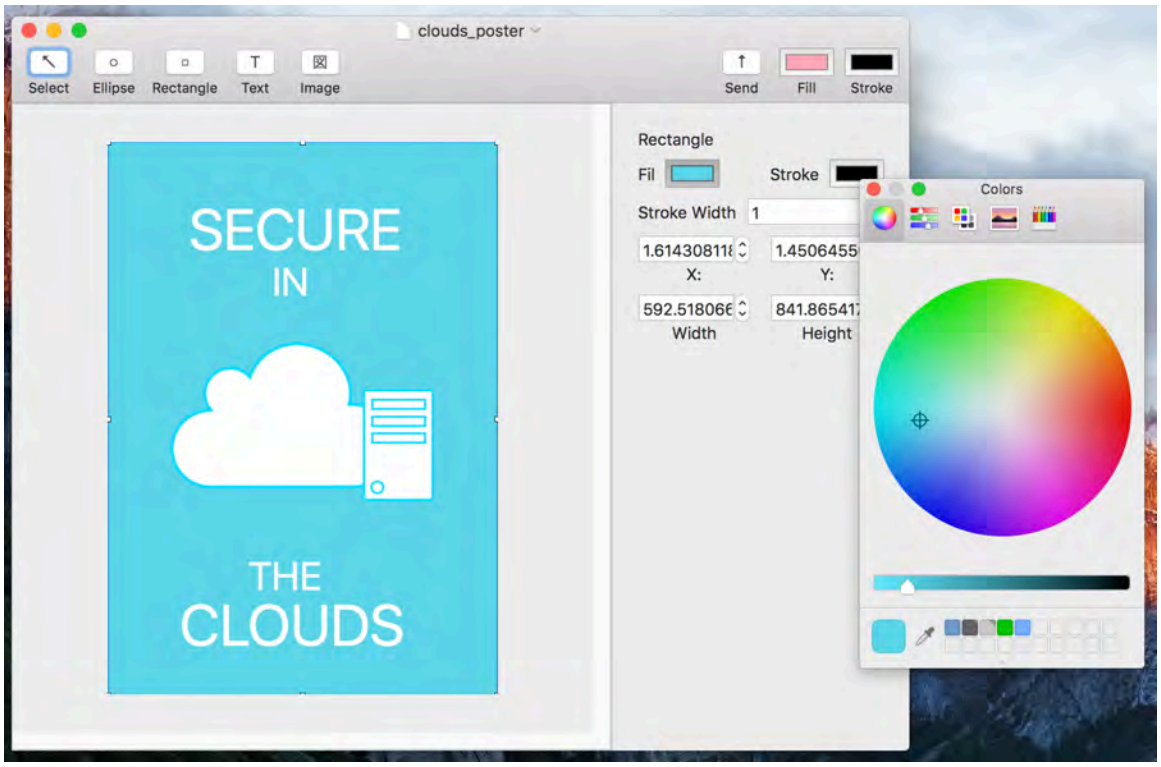

図 5.6: パレット

5.4.2　キャンバス

図 5.4 中の (2) はキャンバスを表す．Base は WYSIWYG インタフェースである．キャンパスはドキュメントそのものを描画しており，ユーザーは選択ツールなどの各種ツールでキャンパス上に表示されているグラフィックを直接編集することで，グラフィックデザインを行える．また，キャンパスはピンチジェスチャによって拡大・縮小が可能である．

5.4.3　インスペクタ

図 5.4 中の (4) はインスペクタである．インスペクタには選択中のグラフィックの詳細が表示される．表示される詳細はグラフィックの種類により異なり，図 5.4 ではテキストグラフィックを選択しているため，範囲及び大きさの他に，フォントやフォントサイズが表示されている．ユーザーは各値を表示しているテキストフィールドに新しい値を打ち込むことにより，正確な値でのグラフィックの配置などを行うことができる．また，テキストフィールドの右に付属するステッパーにより，値の増減が簡単に行える．インスペクタにより変更された値は，キャンバスにリアルタイムに反映される．また，ツールバーの塗りツールと線ツールと同様

のインタフェースで，選択中のグラフィックの塗りと線が表示される．また同様にクリックするとパレットが表示され，選択中のグラフィックの色を変更できる．

5.4.4 ドキュメントの受信

後述する Client から，Relay サーバを介してドキュメントの更新を伝えられると，Mac OS X の通知機能により，更新されたことをユーザに知らせる．図 5.7 に通知により表示されるアラートを示す．アラートはディスプレイ右上に表示される．また，新しいウィンドウに更新されたドキュメントを表示する．通知アラートはユーザーが「閉じる」あるいは「表示」を選択するまで画面右上に残り，ユーザーに更新があった旨を知らせる．

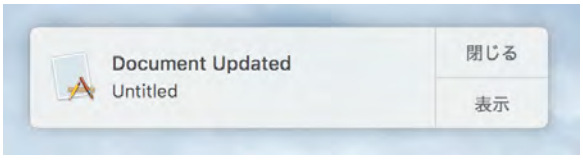

図 5.7: 通知アラート

5.4.5 Adobe Illustrator からの貼り付け

Base は，一般的な DTP ソフトウェアの機能を全て備えてはいない．その代わりに，よく用いられる DTP ソフトウェアである Adobe Illustrator のオブジェクトを画像として貼り付けられる機能を有する．Adobe Illustrator にてオブジェクトを選択し，コピーし，Base にて Edit メニューあるいはショートカットにて貼り付けることで，画像として挿入することができる．図 5.8 に Adobe Illustrator で選択したオブジェクトを Base にて貼り付けている様子を示す．

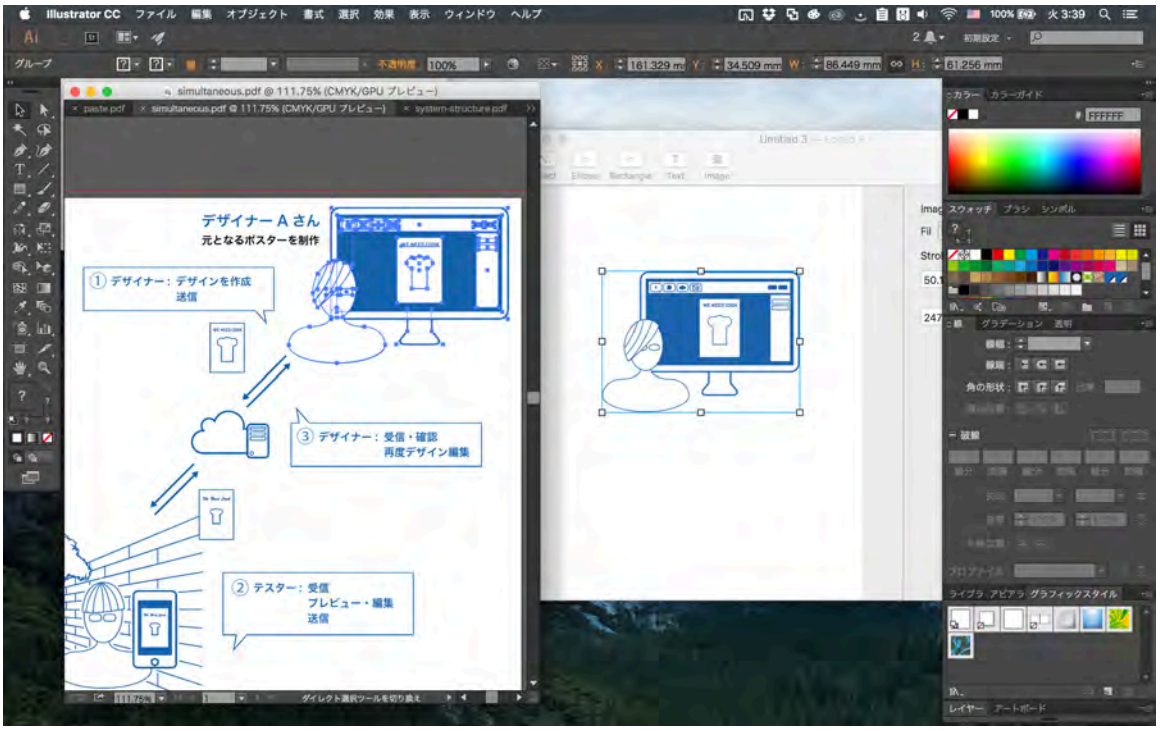

図 5.8: Adobe Illustrator からの貼り付け

5.4.6 メニュー

Base は Mac OS X 用アプリケーションであり，通常の Mac OS X 用アプリケーション同様，メニューから各種の操作が行える．図 5.9 に Base のメニューを示す．図中では例として File メニューを開いている．File メニューからは，ドキュメントの新規作成，保存されているドキュメントを開く，最近のファイルを開く，閉じる，保存する，複製するといった操作が可能である．Edit メニューでは，グラフィックを移動するといった操作のアンドゥ，リドゥ及び各グラフィックのカット，コピー，ペーストなどが行える．Format からはテキストサイズの増減が可能である．Arrange メニューでは，選択しているグラフィックのロック及びロックされたグラフィックのアンロックが行える．ロック状態のグラフィックには，クリック等による選択や移動や拡大・縮小といった一切の操作を行うことはできない．ロック機能は，背景として敷いた図形や画像を，その前にあるグラフィックを操作しようとして誤って移動したり選択したりするのを避けるのに便利であり，Adobe Illustrator 等でも用いられている．Window メニューからはツールバーの表示/非表示やフルスクリーン化が行える．

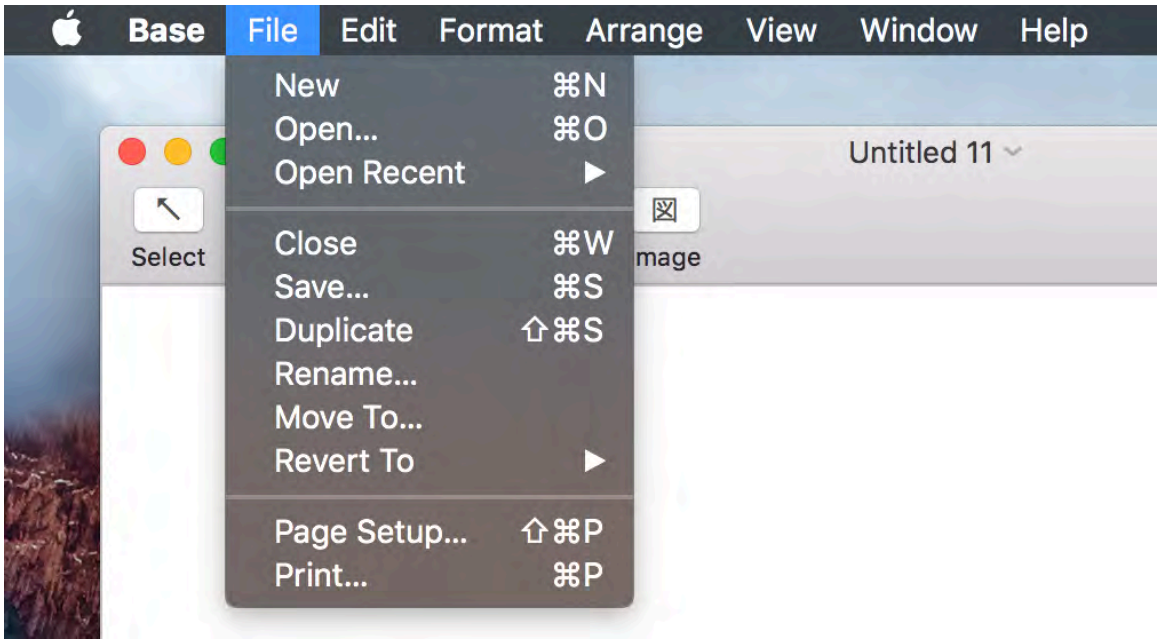

図 5.9: Base のメニュー

5.4.7　ショートカットキー

DTP ソフトウェアでは，各ツールを頻繁に切り替えながら作業を行うため，一般的なソフトウェアでは必ずショートカットキーが備わっている．そこで，Base でもショートカットキーを複数用意した．これにより，マウスで選択中のグラフィックからカーソルを動かすことなく各種ツールに切り替えることでスムーズな作業を行える．表 5.3 にショートカットキーと行える操作を一覧で示す（一般的なソフトウェアと共通の「保存」（Command+S）などを除く）．なお，ショートカットキーとして用いるキーは，可能な限り Adobe Illustrator と共通のものとした．

表 5.3: ショートカットキー一覧

ショートカットキー	操作
v	選択ツール
e	楕円ツール
r または m	矩形ツール
t	テキストツール
i	画像ツール
d	塗りと線を白と黒に戻す

5.5 iOS 用アプリケーション

iOS 用アプリケーション (Client) は，Base より Relay サーバに保存したドキュメントを一覧し，前述した DMAR インタフェースにてデザインのプレビュー・編集が行えるアプリケーションである．

5.5.1 ドキュメント一覧

Client のドキュメント一覧画面を図 5.10 に示す．一覧画面では，Relay サーバに保存されているドキュメント全てが，名前とサムネイル付きで表示される．ドキュメント一覧の更新は，画面を下に引っ張って離す (pull-to-refrseh) 動作によって行える．pull-to-refresh インタフェースは，iOS 純正の「メール」アプリケーションなどにも用いられているインタフェースである．画面を下に引っ張って離し，再読込中の画面を図 5.11 に示す．一覧中のいずれかのサムネイルをタップすることで，DMAR インタフェースによるドキュメントのプレビュー・編集画面へと遷移する．

5.5.2 ドキュメントの色補正

白色領域指定による参考色乗算法

制作したデザインをコンピュータやスマートフォンのディスプレイで見たときの印象と，印刷し実環境で見た場合の印象は大抵の場合において異なる．なぜならば，印刷物の見た目は，それが掲示されている環境に影響されるからである．例えば印刷物が暖色系の照明で照らされている場合は，印刷物も黄色や赤みがかかって見え，また寒色系の照明で照らされている場合は，青みがかって見える．したがって，ただ単にカメラから得られたライブビデオに対して，レンダリングしたポスター画像を重畳表示するだけでは実環境を考慮したプレビュー・編集は行えない．この問題は一般的な拡張現実においても存在し，**光学的整合性**と呼ばれる [1]．AR において，現実物体と仮想物体間の陰影表現や色味が一致しない場合に，重畳された結果に違和感が生じる．例えば，現実空間では画面右奥に光源が存在し，現実物体は左前に影を落としているのにもかかわらず，仮想物体は影を生じていない場合などである．また，前述したように光源の色の問題も存在する．この問題の解決には，カメラを通し，何らかの手法により現実空間の光源や周囲の物体の情報を推定する必要がある．一般に，リアルタイムで現実の情報を推定し仮想物体に反映させる手法としては，広角撮影が可能なカメラを用いて直接光源を測定する手法 [18] と，鏡面球を設置し，それに映り込む実環境をカメラから取得する方法がある．後者の方法において，AR ならではの特徴を活かした手法として，AR マーカーの中心に鏡面球を設置した手法が存在する [19][20].

前者の手法 [18] には広角カメラが必要となり，モバイル端末のみで達成することは難しい．また，後者の手法には，鏡面球及び AR マーカーが必要となる．そこで本研究では，簡易的に照明の色を仮想物体（ポスター）に反映する，**白色領域指定による参考色乗算法**を提案する．

図 5.10: ドキュメント一覧

図 5.11: pull-to-refresh インタフェースにて更新中のドキュメント一覧

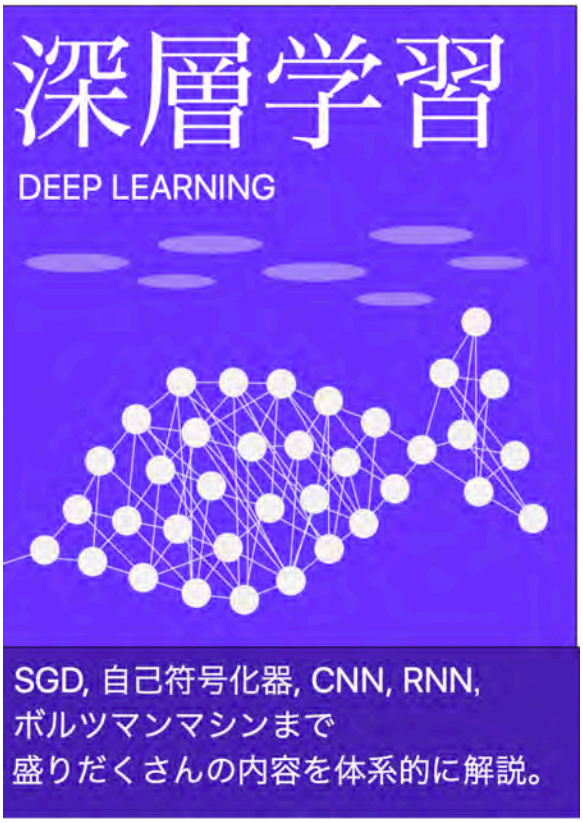

図 5.12: ポスター元画像

まず，仮想物体を重畳表示する前に，ライブビデオをユーザーに提示し，ユーザーはライブビデオ中の，本来は白色であるはずの領域を選択する.「本来は白色である領域」とは，その物体の表面自体が白色であり，白色の光源で一様に照らした場合に「白色」に見える領域，ということである．あるいは，ポスターなどのおいて，印刷前のデータをコンピュータ上で表示した場合に RGB の値が全て最大値となる領域である．実環境において，この領域の色は照明の色の影響を受けているため，一般に完全なる白色ではない．この領域の色を**参考色**として保持する.

次に，仮想物体（本研究の場合，ドキュメントに含まれる各グラフィック）が表す色に，参考色を乗算して描画する．乗算とは，各色チャンネルの値を文字通り掛け合わせる演算であり，例えば真っ白を乗算しても結果に変化はなく，真っ黒を乗算した場合は，結果はすべて真っ黒となる．これにより，各グラフィックにおいて白色の部分は参考色となり，他の部分も同様に色味が変わるため，照明による色味の変化を反映することができる.

図 5.12 にポスターの元画像を，図 5.13 に色を補正していない場合の AR による表示を，図 5.14 に本手法によって色を補正した場合の AR による表示を示す．環境は 5.15 と同様である.

本手法では，照明の色味のみが反映され，現実の陰影の方向などは反映することができないが，本研究で対象としている物体は掲示されているポスターやフライヤーなど厚みがほぼ

図 5.13: 色を補正していない場合の AR 表示

図 5.14: 本手法により色を補正した場合の AR 表示

存在しない物体であり，影を落とすことがないため問題ないと考えられる．一方で，本来は一部分のみ影を受けているといった場合に対しては本手法は用いることができない．本手法の有効性は第 7 章にて評価する．

白色領域指定インタフェース

前述した手法では，本来の白色領域をユーザーが指定する必要がある．本システムでは，DMAR インタフェースにおけるポスターの掲示位置の指定の前に，ユーザーに白色領域の指定を要求するようにした．白色領域指定インタフェースを図 5.15 に示す．この画面において，ユーザーは白色領域を，ライブビデオ中をタップすることにより指定する．指定した色は画面下部に表示され，確認することができる．タップ位置により取得された色が正しい場合は，画面下部の Done ボタンを押すことにより，後述する掲示位置指定インタフェースへと遷移する．誤って白色でない領域をタップしてしまった場合は，再度別の位置をタップすることで，再度指定することができる．

図 5.15: 白色領域指定インタフェース

5.5.3 ドキュメントのプレビュー・編集

Client におけるドキュメントのプレビュー・編集画面では，第 3 章にて述べた DMAR インタフェースにてドキュメントのプレビュー・編集ができるようになっている．本節では，第 3 章にて提案した 4 タスクが Client においてどのように行えるかを記述する．

ポスターの掲示位置の指定

前節にて述べたドキュメント一覧にて，任意のドキュメントをタップにより選択すると，画面が遷移し，ポスターの掲示位置の指定を行う「貼り付け」インタフェースが表示される．図 5.16 に，遷移した貼り付けインタフェースを示す．提案した DMAR インタフェースと同様，ドキュメントが半透明で表示される．また，画面下部のボタンにより任意の位置にポスターを「貼り付ける」ことができる．ポスターが貼り付けられる際は，半透明のドキュメントがアニメーションにより縮小，またフェードアウトする．ドキュメントが掲示位置へ向かい奥行き方向に動いていくように見せることで，ユーザーに,「貼り付ける」行為を体感させる．アニメーションが終了すると，あたかも現実に存在するかのように，掲示位置にドキュメントが描画される．

例として，図 5.17 に，本の表紙を貼り付け対象としている貼り付けインタフェースの画面を，図 5.18，5.19 に貼付け後の画面を示す．これらの図でわかるように，スマートフォンを動かしても，対象の物体を動かしても，貼り付けた物体にドキュメントが描画される．ただし，スマートフォンのカメラ内に貼り付けた対象が存在しない場合は，ドキュメントは描画されず，後述する作業を行うこともできない．

編集するグラフィックの選択

DMAR インタフェースでは，編集するグラフィックはタップで行える．例として，タップにより選択された楕円形グラフィックに，バウンディングボックスが表示されている様子を図 5.20 に示す．選択を解除するには，選択されたグラフィックを再度タップする．各グラフィックをタップをすることで，複数選択が行える．

編集内容の入力

Client では，DMAR インタフェースで提案した編集内容の入力を一通り行うことができる．以下に各操作を説明する．

移動

移動は，選択したグラフィック（複数可）をドラッグすることにより行える．図 5.21 にドラッグによる移動の様子を示す．ユーザーが指でドラッグ操作を行うと，選択したグラフィックは指の動きに追従し，移動する．

図 5.16: Client の「貼り付け」インタフェース

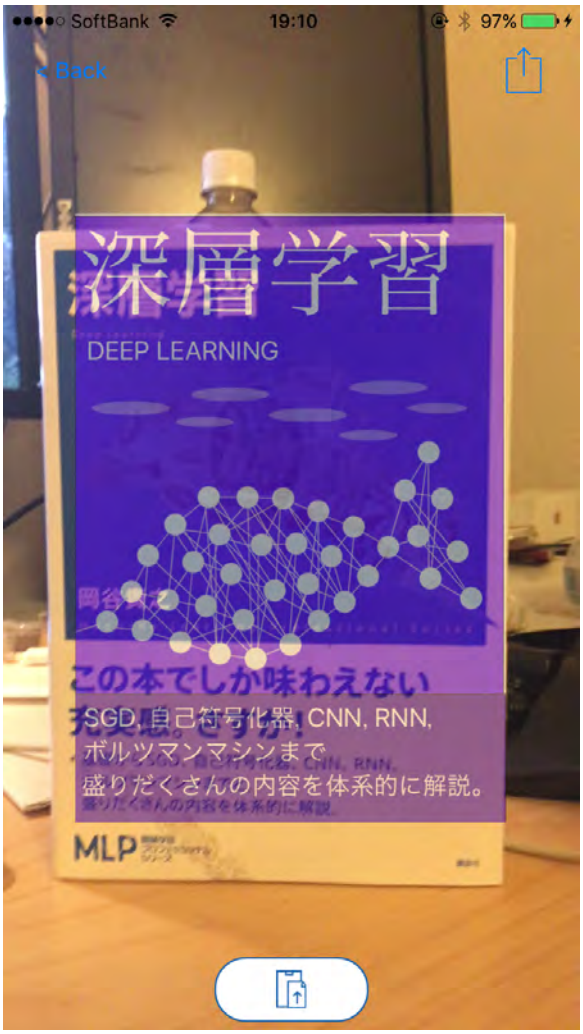

図 5.17: 本の表紙を貼り付ける例

図 5.18: 貼り付け後（斜め上から）

図 5.19: 貼り付け後（本を寝かせた状態）

図 5.20: タップにより楕円形が選択された様子

拡大・縮小

拡大・縮小は，グラフィック（複数可）を選択した状態で，ピンチジェスチャにより行える．ピンチアウトで拡大，ピンチインで縮小する．拡大・縮小する際の基準点は各グラフィックの中心点である．また，何も選択されていない場合は，ドキュメント全体の見え方を拡大・縮小する.「貼り付け」インタフェースにて，小さく／大きく貼り付けてしまった場合の調整に用いる．図 5.22 に，ピンチジェスチャによりグラフィックを拡大した様子を示す.

塗りの変更

塗り色を変更可能なグラフィック（矩形，楕円形，テキスト）を選択すると，図 5.20 中に表示されているように，選択したグラフィックの横にカラーパレットが表示される．このカラーパレット上をタップ・ドラッグすると，選択中のグラフィックの塗り色が，ユーザーの指の下にある色となる．図 5.24 に，パレット上の色が選択中のグラフィックに反映されている様子を示す.

フォントの変更

テキストを選択すると，選択したテキストの横に，様々なフォントによるプレビューが表示される．このフォントプレビューのいずれかをタップすることで，選択されたテキストにフォントが反映される．図 5.23 に，フォントプレビュー及び選択されたテキストにフォントが反映されている様子を示す.

編集結果のフィードバック

上述した編集結果は，ユーザーの操作に合わせ，全てリアルタイムで反映される.

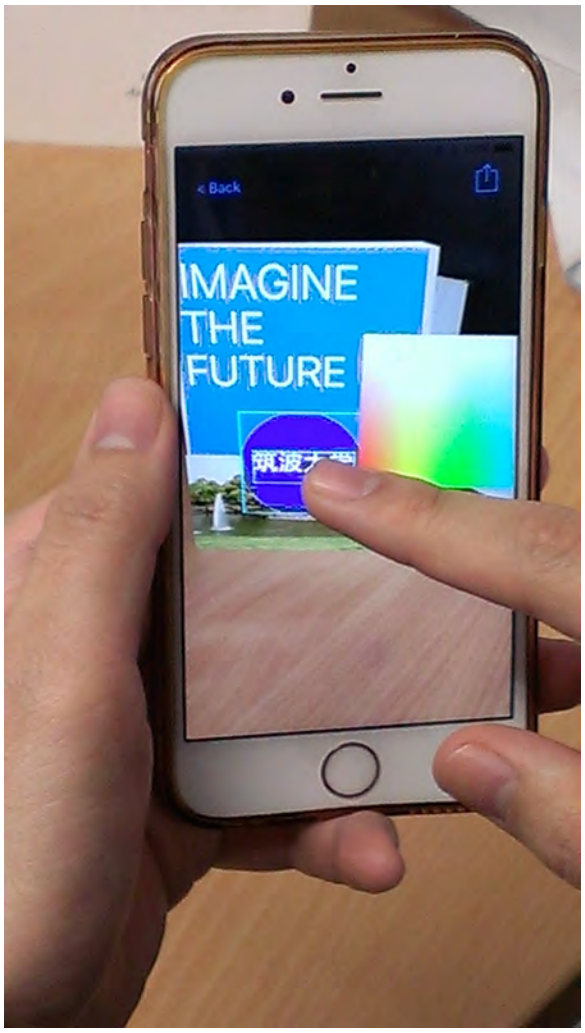

図 5.21: ユーザーのドラッグ操作による移動の様子

図 5.22: ユーザーのピンチ操作による拡大・縮小の様子

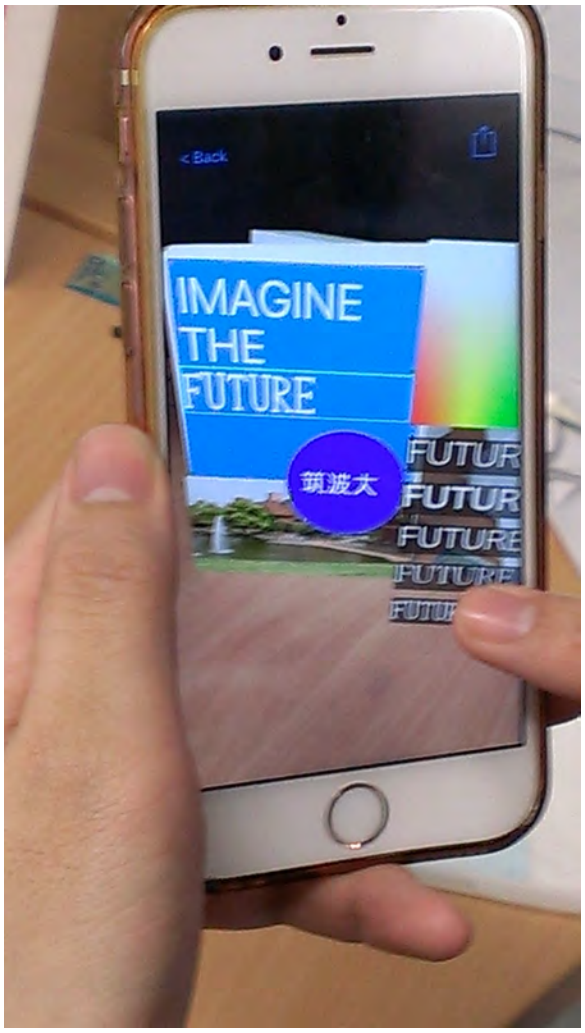

図 5.23: フォントプレビュー及び選択中のテキストにフォントが反映されている様子

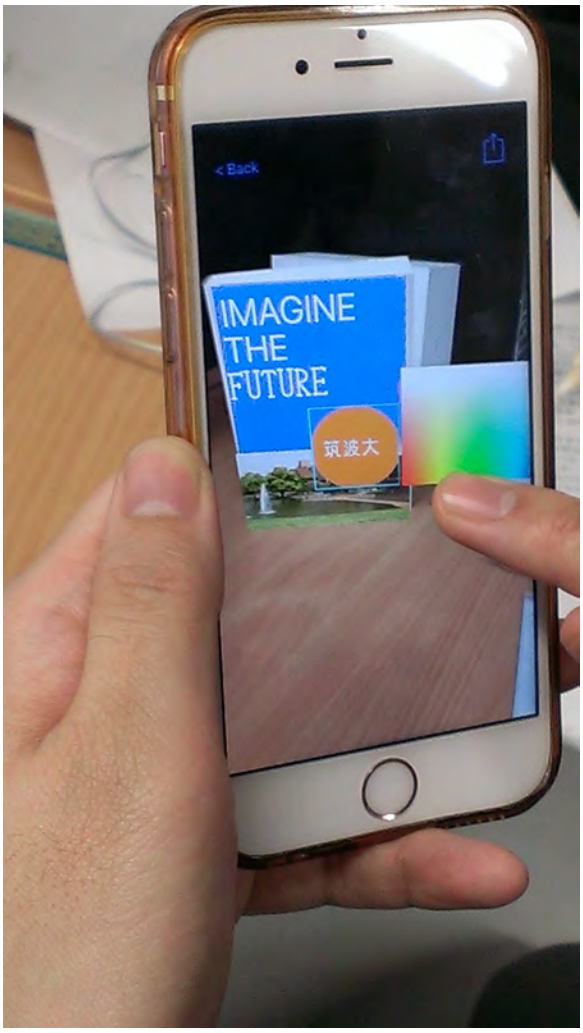

図 5.24: パレット中の色が選択中のグラフィックに反映されている様子

5.5.4 編集したドキュメントの送信

図 5.20 などの右上に表示されている送信ボタンをタップすることにより，編集が反映されたドキュメントを Relay サーバに送信することができる．この際，Client が Relay サーバーにドキュメントを送信したことは，リアルタイムに Base 側に伝えられる．Base 側でのドキュメントの受信については，5.4.4 節にて述べている．送信中は，Base 側と同様アクティビティインジケータが表示され（図 5.25），ユーザーに送信中であることを伝える．

図 5.25: ドキュメント送信中の画面

5.6 ユースケース

本節では，本システムの利用の流れについて記す．本システムの想定ユーザーは，1.2 節にて述べたとおり，コンピュータ側 Base を用い元となるデザインを制作する**デザイナー**と，現地にて Client により実環境を体感しながらのプレビュー・編集を行う**テスター**からなる．デザイナーとテスターが異なり，距離を隔てているがほぼ同時に作業を行う場合を**同時作業**，デザイナーとテスターが同一人物であり，Base にてデザインを行った後，現地に赴き Client でプレビュー・編集を行い，再度コンピュータに戻り，Base にて作業を継続する場合を**逐次作業**と呼称することにする．本節では，この 2 ケースについて，ユースケースを記述する．

5.6.1 同時作業

同時作業の様子を図 5.26 に示す．また，図 5.26 に示されている様子を，時間軸にそって以下に示す．

1. まず，デザイナーである A さんは，大学の掲示板に掲示するポスターの制作を頼まれ，Base にてデザインの制作を行っている．
2. 次に，ラフ案が完成したので，A さんに制作を発注した B さんに「完成したので，一度実環境にて見て来てほしい」と連絡し，A さんは Base からデザインの送信を行う．
3. そこで，テスターである B さんは自らの勤務地である大学の，掲示予定位置の掲示板の前に立ち，Client を起動すると，A さんが制作したデザインがドキュメント一覧に表示されている．
4. B さんはそれを選択し，まず，掲示板中の本来であれば白色の領域をタップする．そこには暖色系の照明が設置されており，本来よりも白色は黄色味がかって見えた．さらに，「貼り付け」インタフェースにより，掲示板の，現在試しているポスターで置き換えられてしまう予定の古いポスターの上に，A さんのデザインを貼ることで，照明の色を考慮したプレビューを行うことができた．そこで，B さんは，この照明ではタイトルが背景色と似通ってしまうことと，タイトルの大きさが，掲示板サイズで確認すると小さいことに気づいた．B さんは色と大きさを Client の DMAR インタフェースにて変更し，より良く見えることを確認した．
5. 最後に，B さんは送信ボタンによりドキュメントを送信した．
6. 仕事場で，他の仕事を行いながら待機していた A さんのコンピュータに通知が届き，Base にて，B さんが修正したバージョンのドキュメントが開かれた．
7. そこで B さんの編集を確認し，再度デザインの修正を行う．
8. 改めて B さんに送信する．
9. B さんは Client にて再度受信する．
10. B さんは現場でプレビューを確認した．後日，実際に印刷し掲示してみると，Client によるプレビューと印象が近く，B さんは満足した．

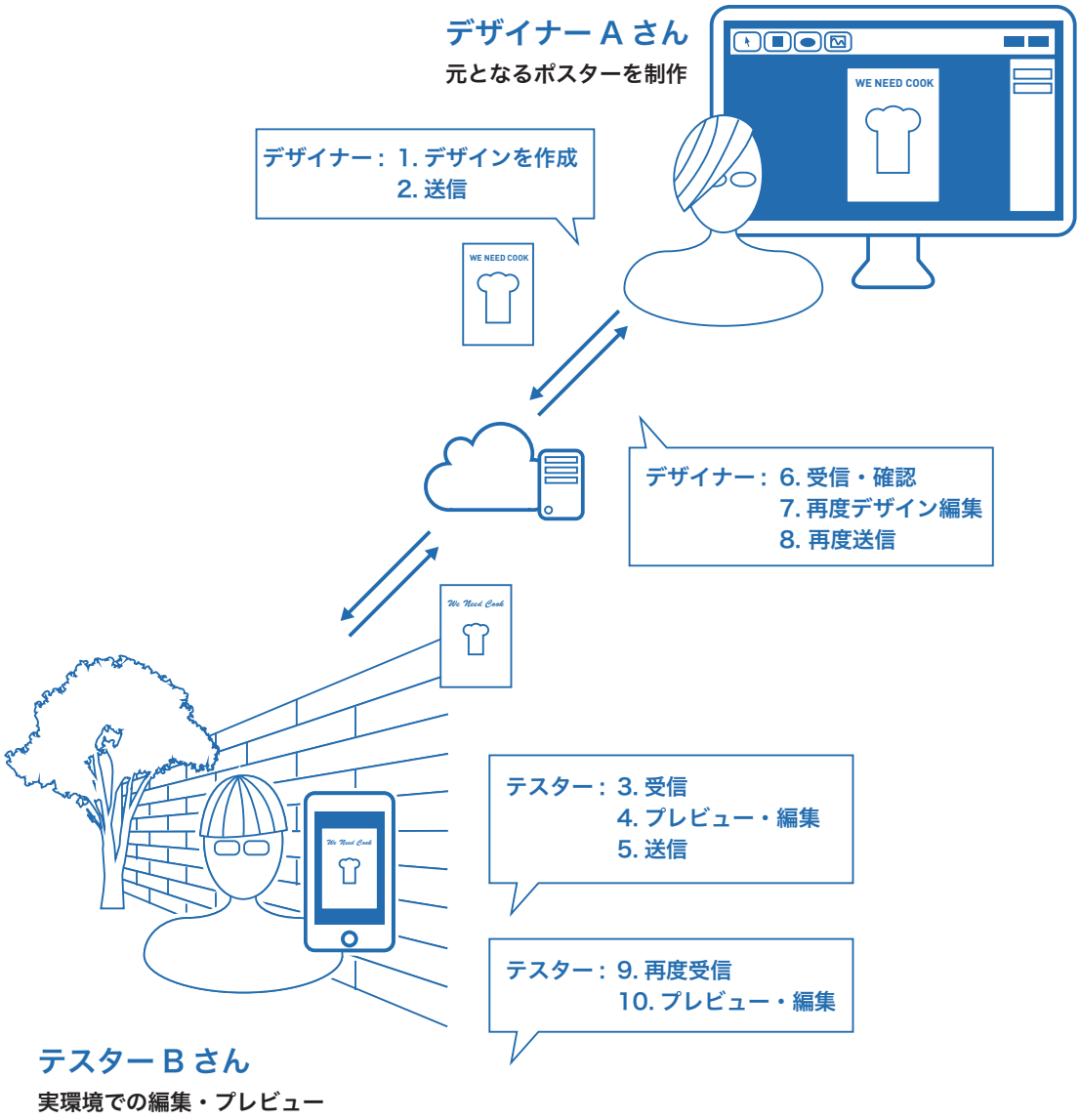

図 5.26: 同時作業の様子

5.6.2 逐次作業

1. 逐次作業の様子を図 5.27 に示す．まず，デザイナーである A さんは，本の表紙のデザイン制作を行っている．

2. ラフ案が完成したので，Base から Relay サーバーに送信する．

3. そして A さんは書店に赴き，その本のジャンルのコーナーの前に立つ．Client を起動し，ドキュメント一覧を更新，制作した本のドキュメントを選択する，

4. 多く並んで置いてある本の一つが本来であれば白色となる色を含んでいたため，その部分をタップし参考色として指定する．次に，その一つの本を手に取り垂直になるようにスマートフォンをかざし,「貼り付け」インタフェースにより「貼り付け」る．すると，その本の表紙は A さんの制作した表紙となり，実際に本の表紙となったらどう見えるか，のプレビューが行えるようになった．A さんはその本を，並んでいた場所に戻し，周りのデザインと比べて目を引きやすいか，埋没していないかを確認した．そこで，A さんはフォントが細いため目立たないことに気づき，タイトルのフォントサイズや位置などを DMAR インタフェースにより編集し，検討する．

5. 色々な試行錯誤の末満足した A さんは，Client から Relay サーバに，編集したドキュメントを送信する．

6. 仕事場に帰った A さんは，コンピュータに届いていた通知から，編集したバージョンのドキュメントを受信.

7. 最後に微調整を行い，本システムにより，他の書籍に埋没しないデザインを作ることができた．

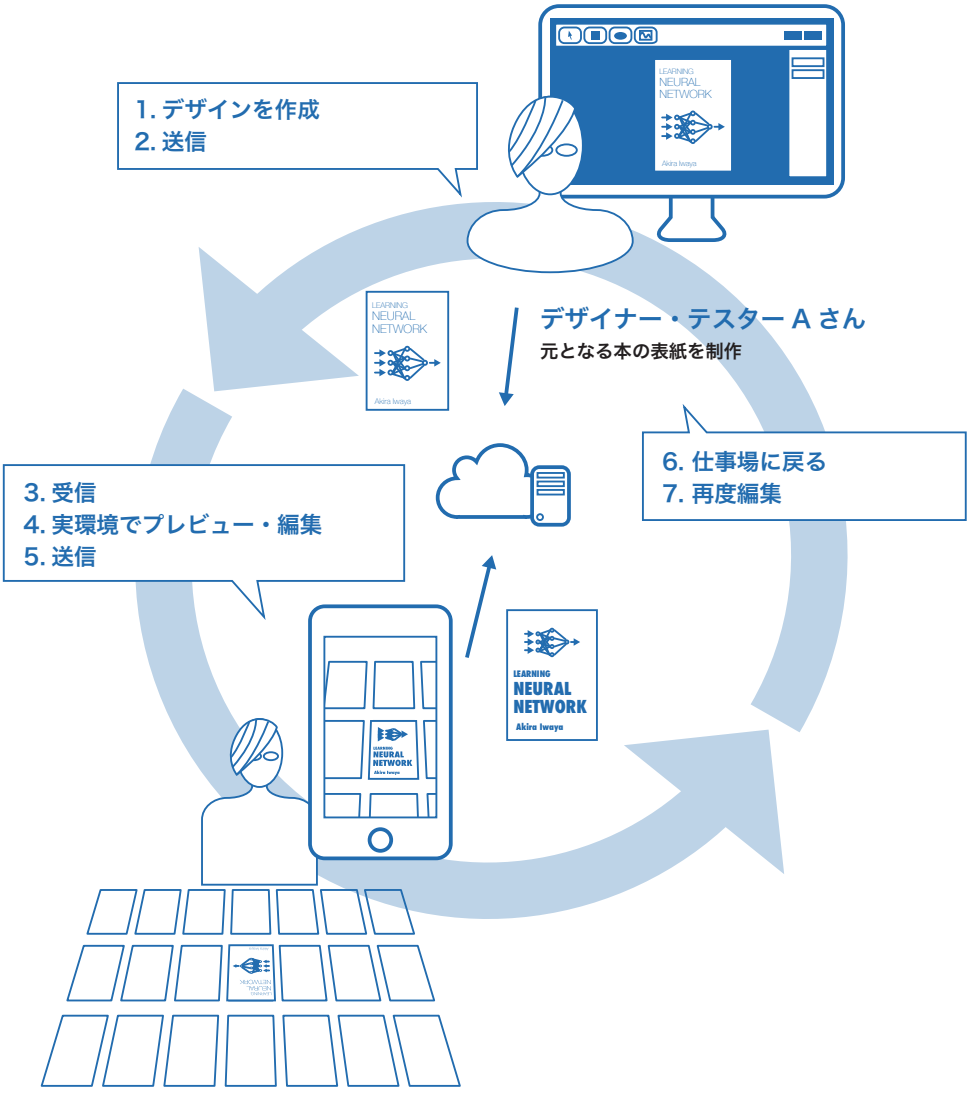

図 5.27: 逐次作業の様子

第6章　仮想ポスターによるデザインシステムの実装

本章では，前章で記述した仮想ポスターによるデザインシステムの実装について詳述する．

6.1　システム構成

本システムの構成を図 6.1 に示す．以下，システムを構成する Base, Relay, Client について順に述べる．

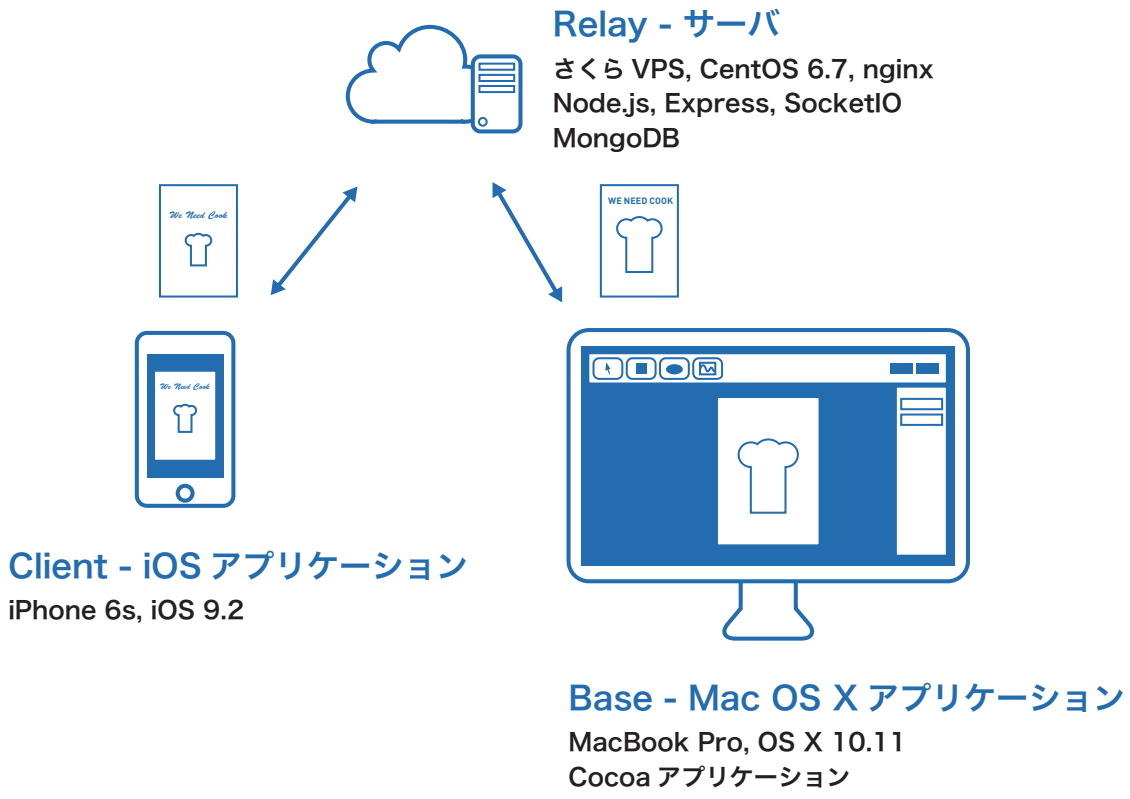

図 6.1: システム構成図

6.2 Mac OS X用アプリケーション

6.2.1 Document-Based Application

Base は Mac OS X用アプリケーションであり，OS X SDK 10.11 及び Swift にて Cocoa（Mac OS X にてアプリケーションを作るためのフレームワーク）アプリケーションとして Xcode で開発した．Base や Apple Keynote，Pages, Preview などのアプリケーションは，Cocoa では Document-Based Application と呼ばれる [21]．Document-Based Application では 1 ウィンドウが 1 つのドキュメントを扱い，複数ドキュメントが複数ウィンドウで開かれることが多い．このようなアプリケーションでは，新規作成，保存など共通する仕組みが多いため，SDK にてそのための API が用意されており，開発者はそれらを用いることができる．ここで言うドキュメントとは，文書やプレゼンテーションなどの構造化されたデータを指す．Base においてもそれらの API を利用している．Document-Based Application では，ドキュメントを表すクラスを，`NSDocument` クラスのサブクラスとして用意すればよい．前章にて定義したドキュメントの構造を保持する `NSDocument` クラスのサブクラスとして，`Document` クラスを作成し，表 6.1 に示すデータをプロパティとして保持するようにした．その他にも，選択中のグラフィックなどの内部状態を有する．表中の `Graphic` 型については，次節にて説明する．

表 6.1: `Document` クラス

プロパティ名	型	説明
identifier	`String`	ID(UUID)
name	`String`	名前
width	`Int`	横幅 [mm]
height	`Int`	ドキュメントの縦幅 [mm]
graphics	`Graphic`	グラフィックの配列

6.2.2 デザインを表すデータ構造

前章で述べた通り，複数のグラフィックという要素でドキュメントを構成し，グラフィックデザインを表している．実装としては，`Graphic` クラスを作成し，そのサブクラスとして矩形クラスや楕円形クラスなどを作成している．`Graphic` クラスに，各グラフィック共通のプロパティとして保持し，その他各グラフィックで特有の属性は各サブクラスにおいて保持している．表 6.2 に `Graphic` クラスのプロパティを示す．表中以外にも，選択されているか，描画しないか，ロックされていないかなどの内部状態を有する．

表中において，`CGRect` は矩形を定義する Cocoa の構造体であり，`NSColor` は色を定義する Cocoa のクラスである．`CGFloat` は Cocoa のグラフィック描画フレームワーク Core Graphics にて定義されている，描画される位置や線などを定義する浮動小数点型である．

表 6.2: Graphic クラスのプロパティ

プロパティ名	型	説明
identifier	String	ID(UUID)
bounds	CGRect	位置とサイズ
fillColor	NSColor	塗り色
strokeColor	NSColor	線色
strokeWidth	CGFloat	線幅

6.2.3 ドキュメントのシリアライゼーション

Base において，ドキュメントは 2 種類の方法にてシリアライゼーションされる．ひとつは，ユーザーが制作したドキュメントを，自身の記憶装置に保存するためのシリアライゼーション，もうひとつは，Relay サーバーに保存し，また Client が受信するためのシリアライゼーションである．

Base では，ドキュメントを File メニューから保存した場合，拡張子 .dmar のファイルとして保存される．これが前者のシリアライゼーション方法である．これには，Cocoa が提供しているシリアライゼーションの仕組みを用いた．Cocoa では，シリアライズしたいオブジェクトにおいて NSCoding プロトコルに準拠し，encodeWithCoder: 及び initWithCoder: メソッドを実装することにより，永続化することができる．詳細は [22] を参照されたい．

一方で，本システムでは Base, Relay, Client で共通してデータを扱わなければならない．そこで，インターネットを介した HTTP 通信による送受信と親和性の高い JSON (JavaScript Orientation Notation) フォーマット [1] でもドキュメントをシリアライズできるよう実装を行った．JSON フォーマットにて表したドキュメントの例を以下に示す．以下で分かる通り，NSColor は red, blue, green, alpha に，CGRect は x, y, width, height として汎用的なフォーマットで表現している．

```
{
  "height" : 297,
  "identifier" : "A8977B29-3CF5-45FD-A277-B41CE61938EF",
  "width" : 210,
  "name" : "deep_learning.dmar",
  "graphics" : [
    {
      "fillColor" : {
        "red" : 0.3407618,
        "alpha" : 1,
        "green" : 0,
        "blue" : 1
      },
      "bounds" : {
        "height" : 841.9138,
        "x" : 0.1345911,
```

[1] JSON の紹介, http://www.json.org/json-ja.html

```
      "width" : 592.4223,
      "y" : 3.405384
    },
    "identifier" : "7F74C689-BDDB-4C0F-9D1C-C570B65E5B1B",
    "strokeWidth" : 1,
    "selected" : 0,
    "previousBounds" : {
      "height" : 0,
      "x" : 0,
      "width" : 0,
      "y" : 0
    },
    "strokeColor" : {
      "red" : 0,
      "alpha" : 1,
      "green" : 0,
      "blue" : 0
    },
    "type" : "Rectangle"
  },
  {
    "fillColor" : {
      "red" : 1,
      "alpha" : 1,
      "green" : 1,
      "blue" : 1
    },
    "type" : "Text",
    "bounds" : {
      "height" : 161.375,
      "x" : 909.9785,
      "width" : 558.8306,
      "y" : 173.4176
    },
    "identifier" : "DBE425C7-57E1-480E-BD7F-8FDB073F6959",
    "strokeWidth" : 1,
    "selected" : 0,
    "text" : 深層学習"",
    "previousBounds" : {
      "height" : 0,
      "x" : 0,
      "width" : 0,
      "y" : 0
    },
    "fontSize" : 130,
    "strokeColor" : {
      "red" : 0,
      "alpha" : 1,
      "green" : 0,
      "blue" : 0
    },
    "fontName" : "HiraginoSans-W6"
  }, ... （以下省略）
```

6.2.4 ドキュメントの描画方法

Base において，ドキュメントの描画は CanvasView クラスが担っている．CanvasView クラスは，Cocoa アプリケーションにおいて画面の一部を表す NSView クラスのサブクラスである．NSView クラスのサブクラスにおいて，自身で描画を行う場合は drawRect: メソッドをオーバーライドし，その中で描画系メソッドを呼び出すことで任意の描画が行える．CanvasView での drawRect:メソッドでは，ドキュメントのグラフィック配列を走査し，各グラフィックの drawContentsInView:メソッドを呼び出す．各グラフィックは drawContentsInView:メソッドを実装し，メソッド内で自身を描画する処理を書く．例えば，矩形を表す Rectangle クラスであれば，NSBezierPath クラスを用い，自身の bounds に合わせ，矩形を描画する．さらに，選択中であればバウンディングボックスをその位置に描画する．この一連の処理を以下に示す．

Algorithm 1 ドキュメント内のグラフィック描画

```
for all graphic in document.graphics such that graphic is hidden do
  graphic.drawContentsInView(canvasView)
  if graphic is selected then
    drawBoundingBox(graphic.bounds)
  end if
end for
```

各描画の処理を，Graphic クラスのサブクラス自身が行うことで，新たなグラフィックの定義を簡単にしている．今後，多角形グラフィックなどを描画したい場合でも，既存の処理を変更することなく Graphic クラスを継承した Polygon クラスを実装するだけで良い．

6.2.5 各種機能の実装

マウスによるインタラクション

Base では，描画されたグラフィックを，マウスによって直接クリックやドラッグすることにより各種編集が行える（第 5 章参照）．この実装では，NSView の mouseDown:, mouseDragged:, mouseUp: といった各種マウスイベントに対応するメソッドを CanvasView にてオーバライドすることで行っている．

ドキュメントの送信

前述した JSON フォーマットのドキュメントを送信する処理では，通信用ライブラリとして Alamofire[2] を用いている．サーバー側の API に対し，multipartFormData としてドキュメントの JSON 文字列及び，ドキュメント中に含まれる画像を TIFF 形式にエンコードし，MIMEType を image/tiff とし PUT メソッドにより送信している．図 5.5

[2] Alamofire, https://github.com/Alamofire/Alamofire

中のアクティビティインジケータの表示には DJProgressHUD_OSX[3] を用いている．サーバーサイドの詳細については 6.3 節にて述べる．

ドキュメントの受信

Client によるドキュメントの更新が発生したことをリアルタイムに検知し，ドキュメントを受信，ユーザーに通知するために，Socket.IO[4] を用いている．Client によってドキュメントが更新されると，`DocumentUpdated` イベントが，そのドキュメントの identifier とともに Relay サーバーを介して，Socket.IO により Base に伝えられる．そこで，`NSUserNotification` を用いて通知を発行し，5.4.4 節にて述べた通知アラートを表示する．

Adobe Illustrator からの貼り付け

Adobe Illustrator は，選択したオブジェクトをコピーすると，外部のアプリケーションに PDF として貼り付けられるようデータを用意する．したがって，Base にてペーストが発生した際に，ペーストボードに PDF データが存在していた場合は一度画像として保存し再度読み込むことで，Adobe Illustrator からの貼り付けを可能としている．

6.3 サーバー

Base と Client の通信に用いる，Relay サーバは仮想専用サーバ (Virtual Private Server; VPS) をレンタルし構築した．サーバのアドレスは現時点で `http://153.126.202.61/` である．OS には CentOS 6.7 を用い，Web サーバとして Nginx[5] 及び Node.js[6] を用いてサーバーサイドアプリケーションを実装した．さらに，RESTful な API を実装するために，Web アプリケーションフレームワークである Express[7] を用いている．データベースは，JSON 形式と親和性が高い MongoDB[8] を用いている．また，Client によるドキュメントの更新をリアルタイムに Base に伝えるために，前述したとおり Socket.IO を用いている．

Base 及び Client によるドキュメントのアップロード・ダウンロードに対応する Web API を定義し実装を行った．RESTful になるような設計を心がけた．6.3 に API 一覧を示す．例えば，`http://153.126.202.61/documents` に GET メソッドにて HTTP リクエストを送信すると，レスポンスとしてドキュメント一覧が JSON で返ってくる．

[3] DJProgressHUD_OSX, `https://github.com/danielmj/DJProgressHUD\_OSX`
[4] Socket.IO, `http://socket.io`
[5] Nginx, `http://nginx.org/en/`
[6] Node.js, `https://nodejs.org/en/`
[7] Express, `http://expressjs.com`
[8] MongoDB, `https://www.mongodb.org`

表 6.3: API エンドポイント，リクエスト及びレスポンス

HTTP メソッド	URL	リクエスト	レスポンス・動作
GET	/documents	-	ドキュメント一覧 (JSON)
GET	/documents/:id	ドキュメントの ID	指定 ID のドキュメント (JSON)
POST	/documents	ドキュメント (JSON) と画像	新規ドキュメント作成
PUT	/documents/:id	ドキュメントの ID・ドキュメント (JSON) と画像	指定 ID のドキュメントを更新
DELETE	/documents/:id	ドキュメントの ID	指定 ID のドキュメントを削除
DELETE	/documents/	-	すべてのドキュメントを削除
GET	/images/:id	画像の ID	指定 ID の画像 (image/tiff)

6.4 iOS 用アプリケーション

Client は，iOS 用アプリケーションである．iOS SDK 9.2 及び Objective-C を用いて Xcode にて実装を行った．端末は iPhone 6s を用いた．本節では簡単にドキュメント一覧及び他の機能について述べた後，AR 部分について詳述する．

6.4.1 ドキュメント一覧

図 5.10 に示したドキュメント一覧では，前述の API を用いドキュメント一覧を JSON にて取得，各ドキュメントに対してサムネイルを作成し，グリッド状に並べている．ドキュメント一覧の取得には，iOS の HTTP 通信を非同期で行えるライブラリである AFNetworking[9] を用いている．またグリッド状にサムネイルを並べるために，iOS SDK の Cocoa Touch の API である `UICollectionViewController` のサブクラスとして `DocumentsCollectionViewController` を実装した．`UICollectionViewController` は，ビューをグリッド状に並べるビューコントローラである．

ドキュメント一覧を取得し，各サムネイルを描画する具体的な流れは次のとおりである：まず，ドキュメント一覧の JSON を取得し，それを元に各 Document オブジェクトを作成，サムネイル画像を生成する．サムネイル画像の生成は，1 と同様の手法による．このタイミングで，`UICollectionView` を更新する．さらに，各サムネイルが画面上に登場し表示されるタイミングで，各ドキュメント中の画像のダウンロードを非同期 [10] で開始する．画像のダウンロードが完了したタイミングで，その画像を含むドキュメントのサムネイルを再度生成し，フェードインアニメーションにより滑らかに更新する．

JSON のダウンロードと画像のダウンロードを別に行っている理由は，例えばドキュメントが 100 個あった場合，後の方のドキュメントはユーザーが画面をスクロールしてはじめて表示される．したがってすべての画像が表示されるかどうかはユーザー次第であるため，すべての画像をダウンロードしてしまうと無駄な通信が発生し，ユーザーを待たせることになってしまうからである．

6.4.2 ドキュメントのプレビュー・編集画面

本節ではドキュメントのプレビュー・編集画面中の各機能について述べる．本システムの中核である AR 部分については，次節にて詳述する．

---

[9] AFNetworking, `https://github.com/AFNetworking/AFNetworking`

[10] 非同期で通信することにより，メインスレッドはブロックされない．iOS ではユーザーインタフェースの描画はメインスレッドで行われるため，メインスレッドにて重い処理を行うとその処理中画面はフリーズしてしまう．

ドキュメントの送信

5.5.4 節にて述べたドキュメントの送信では，Base と同様の処理によって，ドキュメントを更新している．Relay サーバの API の `documents/:id` に更新するドキュメントの identifier 及び更新後のドキュメントの JSON をパラメータとして，PUT リクエストすることによって行っている．また，サーバからドキュメントの更新が完了したレスポンスを受け取ったタイミングで，Socket.IO にて `DocumentUpdated` イベントをサーバに発行する．これを受け取ったサーバは，前述のとおり Base にその更新を伝える．また，図 5.25 中のアクティビティインジケータの表示には，SVProgressHUD[11] を用いている．

色補正のための参考色指定

図 5.15 にて示した色補正のための白色領域指定では，`UITapGestureRecognizer` によりユーザーのタップを検出している．タップを検出した際に，ユーザーのタップ位置の RGB 値を求め，それを画面に表示する．

6.4.3 AR の実装

AR の実装には，**Vuforia**[12] という AR ライブラリ及び，iOS SDK に含まれる 3D グラフィックを描画するフレームワークである **SceneKit**[13] を用いている．Vuforia は Qualcomm が開発した AR 向けモバイルビジョンプラットフォームである．通常の AR アプリケーションの実装には，ARToolKit[23] や Unity[14] などが多く用いられるが，前者では，仮想物体の描画に直接 OpenGL を用いなければならない点，後者では AR 以外の画面において通常の iOS のインタフェースを提供できない点などから，このような選定とした．なお，Vuforia と SceneKit を組み合わせ拡張現実を実装する方法に関して詳述する資料はインターネット上に見つからず，本論文が初となることが期待できる．なおこの方法については，筆者が簡易的に手順を示した記事をすでにインターネット上に公開済みである [24]．本節では，まず AR におけるマーカーの 3 次元位置推定手法について述べた後，実際に Vuforia と SceneKit にて実装する方法を詳述する．

AR におけるマーカーの 3 次元位置推定

本節では，AR の基本であるマーカー AR において，カメラで撮影した 2 次元画像からマーカーの 3 次元空間中の位置・回転を推定する手法について述べる．マーカー AR とは,「マーカー」と呼ばれる，特徴的な図形パターン上に仮想物体を重畳表示する AR のことである．本システムにおけるマーカーは,「貼り付け」インタフェースにより指定した領域となる．マー

[11] SVDProgressHUD, `https://github.com/TransitApp/SVProgressHUD`
[12] Vuforia, `https://developer.vuforia.com`
[13] SceneKit, `https://developer.apple.com/scenekit/`
[14] Unity, `http://unity3d.com/jp/`

カー上に仮想物体を重畳表示するためには，カメラで撮影した 2 次元の画像中に写っているマーカーが，現実の空間中のカメラに対して相対的にどこに存在するかという 3 次元の座標を復元する処理を行う．この処理は次のような流れとなる．

1. マーカー座標系（世界座標系とも言う．マーカーの位置を原点とした 3 次元座標系.）を，カメラ座標系（カメラを原点とした 3 次元座標系）に変換する．つまりカメラから見たマーカーの相対的な位置を求める．

2. カメラ座標系（3 次元）から，最終的にユーザーに提示する 2 次元の座標系に変換する．

この処理を，同次座標系を用いて数式で表すと式 6.1 となる．この式により，マーカー座標系の点を撮影画像上の点に変換できる．

$$\lambda \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_{c0} \\ 0 & f_y & y_{c0} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t_x \\ r21 & r22 & r23 & t_y \\ r31 & r32 & r33 & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{6.1}$$

x_c, y_c は，最終的に得られる撮影画像上の点である．λ は，3 次元座標系の点を 2 次元座標系に変換したために生じる，奥行きを表す自由度である．f_x, f_y はカメラの焦点距離であり，x_{c0}, y_{c0} は撮影画像の中心の点を示す．理論的には撮影画像の横幅，縦幅の半分の値と一致する．r_{ij} は回転成分 $\boldsymbol{r}$ を，t_x, t_y, t_z は並進成分 $\boldsymbol{t}$ を表す．X_w, Y_w, Z_w はマーカー座標系の点である．左辺の左から 1 つ目の行列を**内部パラメータ行列**，2 つ目の行列を**外部パラメータ行列**という．より正確には，さらにレンズ歪み行列という，カメラレンズの歪みを表した行列がさらに乗算されるが，最近のカメラのレンズはほとんど歪んでいない場合が多く，無視しても違和感なく合成できる場合が多い．

内部パラメータ行列 (instrinc parameter matrix) はカメラの焦点距離などの情報を表している．カメラに固有であり，一度求めた後は，同じ種類のカメラであれば使い回すことができる（e.g. iPhone 6s のカメラの内部パラメータはすべての iPhone 6s で共通である）．Vuforia にはあらかじめ様々なカメラの内部パラメータ行列が登録されている．この内部パラメータ行列が適切でないと，カメラから入力される画像やライブビデオと仮想物体を描画する際の焦点距離があわず，結果として遠近感が異なって，浮いたような見え方になってしまう．未知のカメラの内部パラメータ行列を求める方法としては，**Zhang の方法**という手法が有名である [25].

外部パラメータ行列 (extrinsic parameter matrix) は，座標系の変換を表す行列であり，回転成分 $\boldsymbol{r}$ と並進成分 $\boldsymbol{t}$ からなる．この行列により回転および平行移動を表現することができ，ある点に回転および平行移動を加え別の点に写した時の関係を記述することができる．このため，マーカー座標系における，ある点がカメラ座標系からするとどこにあるのかを求めることができる．外部パラメータ行列は，ユーザーがカメラをどう保持しているか，マーカーがどこに置かれているかによって変わるため，ライブビデオで AR を行う際は毎フレーム求め

る必要がある．理論的には，外部パラメータ行列は **PnP (Perspective n-Point)** 問題を解くことによって得ることができる [26]．マーカーを平面だと仮定すると，3 次元座標中の同一平面上の 4 点と，その点群が変換後の 2 次元画像中ではどこに位置するかという対応と，内部パラメータ行列を用いることによって，他の 3 次元中の点は 2 次元画像中ではどこに対応するかを推定する問題である．

一般的な AR の実装

一般的には，毎フレーム上記の行列の演算を行い 2 次元の座標にまで変換することはほとんど行われない．その代わり，3D グラフィックスライブラリに上記のパラメータを元にして適切な値を入力することで，ライブビデオの映像と，仮想物体のレンダリングを一致させる．OpenGL では，内部パラメータ行列を元に透視投影行列を，外部パラメータ行列を元にモデルビュー行列を設定すれば，現実でのカメラとマーカーの関係を，レンダリングする仮想空間内でそのまま再現することができる．Vuforia の iOS 向けのサンプル [15] では OpenGL を用いているため，まさにこのような処理を行っている．サンプル中の UserDefinedTarget というサンプルでは，ユーザーが撮影した画像に写っている特徴的な矩形領域をマーカーとして AR を行うことが可能であり，本研究ではこのサンプルを元に開発を行った．サンプル中の `UserDefinedTargetsEAGLView` クラスの `renderFrameQCA` メソッド中で，Vuforia が求めた外部パラメータからモデルビュー行列を取得している処理は

```
QCAR::Matrix44F modelViewMatrix =
QCAR::Tool::convertPose2GLMatrix(result->getPose());
```

である．また透視投影行列は

```
&vapp.projectionMatrix.data[0]
```

に保存されている．

6.4.4 AR によるインタラクション

本システムでは SceneKit を 3D グラフィックスライブラリとして用いている．SceneKit を採用した理由は，iOS の各種フレームワークと親和性が高く，3D 空間をシーン, 各 3D オブジェクトをノードとして表すことにより，複雑な仮想空間を簡便に作成することができるからである．

SceneKit におけるカメラ

式 6.1 や Vuforia から得られる外部パラメータ行列は，物体をカメラから見た座標に変換する行列であり，OpenGL ではそのままモデルビュー行列として用いることができる．しかし，

[15] Vuforia Developer Portal, `https://developer.vuforia.com/downloads/samples`

SceneKit ではモデルビュー行列といった概念は存在せず，SCNCamera を持つノード（ノードは SCNNode クラスのインスタンスとして表される）の位置や回転を設定することにより，カメラの位置やどこを向いているかを決定する．そのため，マーカー座標系を基準とした時に，カメラがどこに存在するのかをノードに設定しなければならない．そのため，外部パラメータ行列の逆行列を求める必要がある．具体的には，下記に示す処理により逆行列を SCNCamera に設定した．また，内部パラメータ行列から透視投影行列を設定するには SCNCamera の projectionTransform プロパティに設定すればよい．

```
// の行列をの行列に変換するメソッドVuforiaSceneKit
- (SCNMatrix4)SCNMatrix4FromQCARMatrix44:(QCAR::Matrix44F)matrix {
    GLKMatrix4 glkMatrix;

    for(int i=0; i<16; i++) {
        glkMatrix.m[i] = matrix.data[i];
    }

    return SCNMatrix4FromGLKMatrix4(glkMatrix);

}

// 逆行列を計算してに設定するメソッドcameraNode
- (void)setCameraMatrix:(QCAR::Matrix44F)matrix {
    SCNMatrix4 extrinsic = [self SCNMatrix4FromQCARMatrix44:matrix];
    SCNMatrix4 inverted = SCNMatrix4Invert(extrinsic); // 逆行列
    self.cameraNode.transform = inverted; // カメラのノードのにセットtransform
}
```

最終的にカメラから見たシーンをレンダリングし，画面に表示する処理は通常 SCNView が担う．しかし，今回は Vuforia が OpenGL にて描画したライブビデオに重畳して表示する必要があるため，SceneKit のレンダリング結果を OpenGL コンテキスト内で描画することのできる SCNRenderer を利用した．これで，後は通常通りシーンに対して 3D オブジェクトを追加することで，マーカーの位置に 3D オブジェクトを重畳してレンダリングすることができる．

AR によるドキュメントの表示

3D 空間内でドキュメントを描画しドキュメント中のグラフィックとのインタラクションを可能にするために，各グラフィックをノードとしてシーンに追加した．SceneKit のノードには，大別して空ノードと，ジオメトリを有するノードがある．ジオメトリは 3D オブジェクトのポリゴンを表すものであり，あらかじめ平面や立方体，球などのプリミティブな形状が用意されている．そこで，Base にて作成された各グラフィックのサイズの平面ジオメトリを作成し，グラフィックを画像として描画したものをジオメトリのテクスチャとして設定することで，ドキュメントを表示する．この際，全てのグラフィックノードの親として空ノードを原点に配置する．以降，このノードを**ドキュメントノード**と呼称する．原点に配置することにより,「貼り付け」インタフェースで指定した位置の中央に描画されることになる．

さらに，5.5.2 節にて述べた参考色乗算法は，指定された参考色をジオメトリのマテリアルの `multiply` プロパティに設定することで実装した．これにより参考色が乗算されて描画される．乗算と元のテクスチャは別の処理として行われるため，グラフィックの塗り色を変更しても，リアルタイムに参考色が乗算された状態のグラフィックをプレビューできる．

触れられたグラフィックの判定

5.5.3 節で述べたような，ユーザーのタッチパネルへの入力による選択などの処理を実現するためには，ユーザーがタップやドラッグを行った画面上の位置座標を，3 次元空間に変換する処理が必要である．単なるタップであれば，`SCNRenderer` の `hitTest:options:` メソッドにユーザーがタップした位置を引数として渡すことにより，そのタップ位置を始点として画面奥方向に伸びる半直線と交差するノードを取得することができる．一方で，選択したグラフィックの移動など，ドラッグが含まれる操作では，やや複雑な手順となる．

まず，ドラッグの検出には `UIPanGestureRecognizer` を用いる．これにより，ユーザーによるドラッグ開始，ドラッグ中を検出できる．各タイミングにおける処理を示す．なお上付き矢印で記されている変数は，3 次元ベクトルであり，SceneKit では `SCNVector3` という構造体で表現される．

1. ドラッグ開始

選択されたグラフィックの 3 次元座標を `SCNRenderer` の `projectPoint:` メソッドにより，レンダリングされた 2 次元座標系ではどこに位置するかを求め，保存する ($\overrightarrow{projectedPoint}$)．次に，ユーザーがタップした点（2 次元座標系）を `unprojectPoint:` メソッドにより，3 次元座標へと変換する ($\overrightarrow{unprojectedPoint}$)．この際，奥行き ($z$) 座標は $\overrightarrow{projectedPoint}$ の z 座標とする．さらに，$\overrightarrow{unprojectedPoint}$ は，ドキュメントノードのローカル座標に変換し，保存する（$\overrightarrow{previousUnprojectedPoint}$）．

2. ドラッグ中

再度，ユーザーのタップした点を `unprojectPoint:` メソッドにより 3 次元座標へと変換する．この際も z 座標は $\overrightarrow{projectedPoint}$ の z 座標とする．そして，ドキュメントノードのローカル座標に変換する ($\overrightarrow{unprojectedPoint}$)．次に，$\overrightarrow{unprojectedPoint}$ と $\overrightarrow{previousUnprojectedPoint}$ の差を求める ($\overrightarrow{delta}$)．そして，選択されたグラフィックノードの位置ベクトル（`position` プロパティ）に $\overrightarrow{delta}$ を加えることで，移動する．最後に，$\overrightarrow{previousUnprojectedPoint}$ を $\overrightarrow{unprojectedPoint}$ にて更新する．ドラッグ中の処理は，ユーザーが指を動かすたびに呼び出されるため，ユーザーの指移動に応じてグラフィックも移動する．

DMAR インタフェースにて定義した他の操作について，以下に実装の概要を示す．

タップによる選択

前述した手法にて，タップされたグラフィックノードを検出し，そのノードの平面ジ

オメトリと同じサイズの平面ジオメトリを含むノードを作成する．そして，その平面ジオメトリのテクスチャとして，輪郭線を描画した矩形を割り当てる．そのノードを，タップされたノードの子ノードとして追加することで，図 5.20 に示すような，選択された見た目を実装している．

拡大・縮小

拡大・縮小には，ピンチジェスチャを検知する `UIPinchGestureRecognizer` を用いてピンチジェスチャの拡大率を取得し，その値をノードの `scale` プロパティに乗じることで拡大・縮小を行っている．また，`scale` プロパティにの変化に合わせて，ノードに対応する `Graphic` オブジェクトの `bounds` プロパティも変更していることで，見た目とモデルオブジェクトの値を一致させている．

塗りの変更

塗りを変更可能なグラフィック（矩形，楕円形，テキスト）を選択すると，図 5.24 に示すようにパレットが表示される．これは選択されたノードの位置に合わせて，パレットの画像を設定した平面ジオメトリを含むノード（パレットノード）を子ノードとして追加することにより行っている．さらに，タップやドラッグされたノードがパレットノードだった場合には，次式によりタップされた位置の色を HSB 空間にて求め，選択されたノードに反映している．

$$hue = \frac{p_x + p_{width}/2.0}{p_{width}} \tag{6.2}$$

$$saturation = 1.0 - \frac{p_y + p_{height}/2.0}{p_{height}} \tag{6.3}$$

$$brightness = 1.0 \tag{6.4}$$

ここで，p はパレットノードのローカル座標, p_{width}, p_{height} はパレットのサイズである．彩度の計算の際に 1.0 から減算を行っているのは，SceneKit での座標系では上方向が y 軸のプラス方向だからである．

フォントの変更

テキストを選択すると，図 5.23 に示すように，パレットともにフォントの一覧が表示される．表示手法は前述のパレットと同様であるが，選択されたテキストに合わせてリアルタイムでフォントのプレビューを生成し，平面ジオメトリのテクスチャとしている．フォントプレビューのいずれかをタップされた際に，そのフォントにて選択されたテキストグラフィックを更新，再描画を行っている．

第7章　評価

本章では，提案システムの有効性を明らかにするために行った各種実験の内容及び結果を述べ，考察する．

7.1　実験 1: 白色領域指定による参考色乗算法の有効性検証

5.5.2 節にて提案した，照明色を仮想物体に反映する手法である白色領域指定による参考色乗算法の有効性を検証するため，様々な照明色の環境下において，AR にて描画した仮想物体の色と現実の色の見え方が近くなるかの検証を行う．まず，図 7.1 に示すマンセル表色系の基本色+中間色を元とした計 10 個の色及び灰色を含むポスター（色見本ポスター）を印刷し，掲示する（図 7.2）．マンセル表色系の RGB 値での表現では，Adobe 社の Web サイトの図を元とした [1]．次に，色見本ポスターを，Client アプリケーションにて，実際に掲示した色見本ポスターの位置に「貼り付け」，参考色を乗算した状態での見え方が，実際の色見本ポスターにどの程度近いかを環境の照明色を変えながら比較する．照明には，1600 万を超える色を発することのできる LED 電球の hue[2] を用い，白，黄，赤，青，緑の 5 色を照明色とする．見え方の比較においては，実際に印刷した色見本ポスター及び Client アプリケーションにて AR にて参考色を乗算し描画した仮想色見本ポスター中の各色を RGB 値として抽出し，比較を行う．色見本ポスター中の灰色及び 5R（1 番上の赤色）の中間の白色領域を参考色として指定した．

7.1.1　結果

実験 1 の結果を，図 7.3 に示す．図中では，照明色 5 色にて照らした，実際の色見本ポスター中の色 11 色の RGB 値及び，AR にて参考色を乗算し描画した色見本ポスター中の色 11 色の RGB 値と，対応する色の RGB 値の差（絶対値）が示されている．RGB の値は 0〜255 である．また各セルの背景色は，その RGB 値の色である．またこの際の参考色を図 7.4 に，色見本ポスターの実際の RGB 値を図 7.5 に示す．また，各照明食において参考色を決定した際の白色領域指定インタフェースの画面を図 7.6〜7.14 に，その際の参考色が乗算された AR による仮想色見本ポスターのプレビューを図 7.7〜7.15 に示す．

[1]The Munsell Color System - Color Models - Technical Guides, http://www.adobe.com/jp/support/techguides/color/colormodels/munsell.html

[2]Meet hue,http://www2.meethue.com/ja-jp/

図 7.1: 色見本ポスター

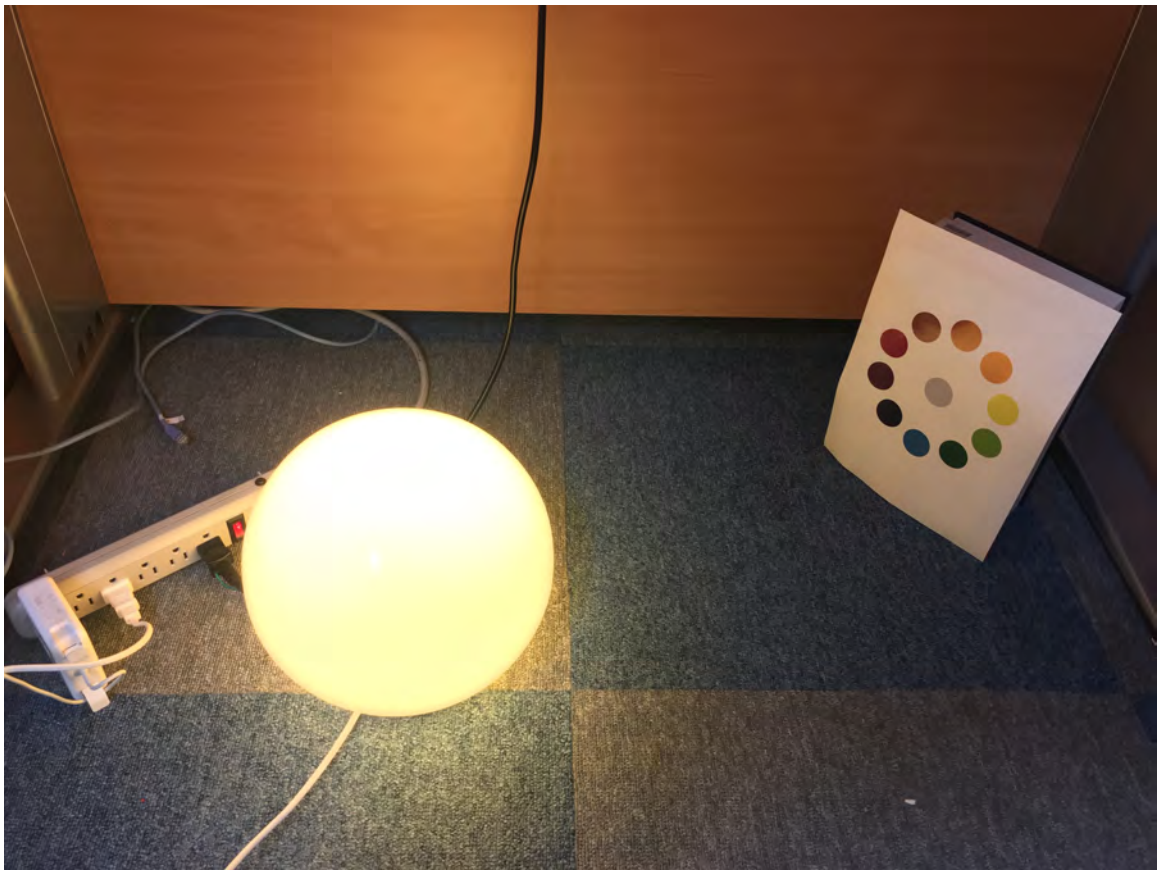

図 7.2: 色見本ポスターを実験環境に掲示した様子

	色	5R			5YR			5Y			5GY			5G			5BG			5B			5PB			5P			5RP			GRAY		
	照明色	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
実際の色	白	79	59	68	105	80	57	131	142	55	57	86	32	15	35	23	19	39	81	14	14	26	35	24	40	79	54	77	91	84	100	104	110	123
	黄	85	38	23	80	32	16	125	96	92	44	44	18	22	23	14	21	11	12	21	11	12	40	16	13	90	30	25	89	44	27	91	67	60
	赤	108	2	13	114	20	19	123	69	28	56	38	15	26	14	14	32	19	38	24	3	12	56	1	10	107	2	18	97	2	14	108	47	62
	青	40	16	146	64	10	100	85	66	78	39	35	50	15	10	44	11	13	130	8	2	54	21	7	95	40	14	134	30	19	156	67	27	152
	緑	72	72	40	77	65	34	91	116	49	35	69	28	14	36	23	14	43	55	48	36	20	13	20	16	39	45	30	66	77	40	78	102	89
参考色を乗算したARの色	白	99	20	24	130	90	24	144	142	39	80	109	30	11	69	36	17	81	108	14	20	36	58	19	46	93	15	55	79	19	40	121	127	128
	黄	88	13	14	116	57	15	127	89	23	69	68	18	9	44	19	11	51	53	13	12	18	52	13	23	82	10	29	70	12	21	107	81	65
	赤	109	12	18	143	47	19	156	74	27	85	57	20	9	36	23	14	42	68	16	10	23	65	11	30	104	10	37	88	11	27	134	68	83
	青	54	4	27	71	13	16	78	21	31	42	15	28	4	11	41	9	21	128	8	5	43	32	5	55	51	5	65	43	4	47	67	27	151
	緑	80	20	20	104	87	21	114	138	34	64	107	27	10	68	30	15	80	87	12	19	30	49	20	38	77	15	46	65	18	33	99	125	105
差	白	20	39	44	25	10	33	13	0	16	23	23	2	4	34	13	2	42	27	0	6	10	23	5	6	14	39	22	12	65	60	17	17	5
	黄	3	25	9	36	25	1	2	7	69	25	24	0	13	21	5	10	40	41	8	1	6	12	3	10	8	20	4	19	32	6	16	14	5
	赤	1	10	5	29	27	0	33	5	1	29	19	5	17	22	9	18	23	30	8	7	11	9	10	20	3	8	19	9	9	13	26	21	21
	青	14	12	119	7	3	84	7	45	47	3	20	22	11	1	3	2	8	2	0	3	11	11	2	40	11	9	69	13	15	109	0	0	1
	緑	8	52	20	27	22	13	23	22	15	29	38	1	4	32	7	1	37	32	36	17	10	36	0	22	38	30	16	1	59	7	21	23	16

図 7.3: 実験 1 結果

照明色	参考色 R	G	B
白	149	156	158
黄	138	103	82
赤	107	43	57
青	88	23	204
緑	112	144	118

図 7.4: 実験 1 参考色

色名	色 R	G	B
5R	169	33	41
5YR	221	143	41
5Y	242	228	64
5GY	136	179	52
5G	15	117	61
5BG	38	140	183
5B	26	34	64
5PB	107	34	80
5P	168	27	95
5RP	138	31	67
GRAY	214	214	214

図 7.5: 実験 1 色見本ポスター中の 11 色

7.1.2 考察

図 7.3 および図 7.7〜7.15 から，本手法はどの照明色においても，AR で描画したポスターに照明色をおおむね反映できていることがわかる．図 7.3 において，最も正確に色を反映出来た条件は，照明色が青の際の灰色 (GLAY) であり，RGB 値の B が 1 異なっていたのみであった．他にも，照明色が黄の際の群青色 (5B) などでも RGB 値の差の合計はわずか 15 であり，人間の目にはほぼ同じ色に見えると考えられる．全体的に，元の色と著しく異なる結果を生じたものは少なかった．最も差が大きい条件では，照明光が青のときの赤紫色 (5RP) のときで，RGB 値の差の合計は 137 であった．また，対照例として，本手法を用いない時に，照明光を赤とし，同様の環境における色見本ポスターの AR によるプレビューを図 7.16 に示す. 図 7.16 と図 7.11 を比較すると，図 7.16 の方が違和感が大きいことがわかる． 一方で，実際の色見本ポスターは，両端などにおいて，照明の当たり方が異なるため，ポスター中の領域によって色が変わる部分があり，このような状態には本手法は対応できないが，簡易に行える手法として一定の有効性を確認できた．

図 7.6: 実験 1 白色領域指定（照明色：白）

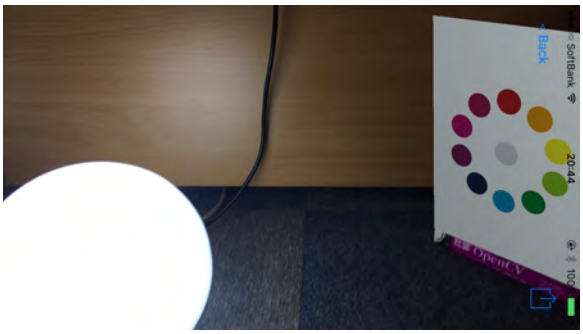

図 7.7: 実験 1 色見本ポスター（照明色：白）

図 7.8: 実験 1 白色領域指定（照明色：黄）

図 7.9: 実験 1 色見本ポスター（照明色：黄）

図 7.10: 実験 1 白色領域指定（照明色：赤）

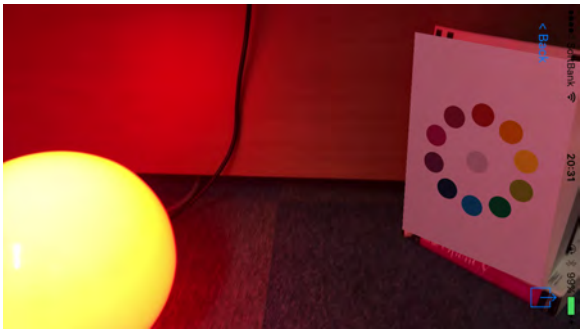

図 7.11: 実験 1 色見本ポスター（照明色：赤）

図 7.12: 実験 1 白色領域指定（照明色：青）

図 7.13: 実験 1 色見本ポスター（照明色：青）

図 7.14: 実験 1 白色領域指定（照明色：緑）

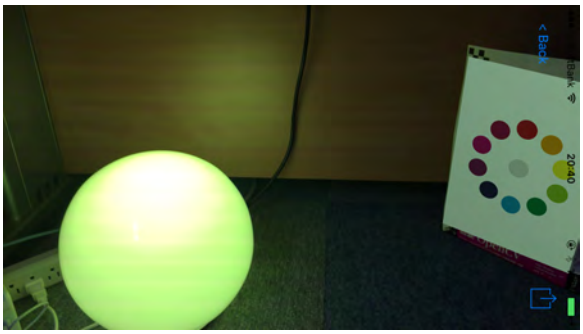

図 7.15: 実験 1 色見本ポスター（照明色：緑）

図 7.16: 参考色を乗算しない場合の色見本ポスターのプレビュー

7.2 実験 2: システムの有用性についてのアンケート調査

第 5 章にて述べた仮想ポスターによるデザインシステムの有用性を評価するため，アンケート調査を行う．被験者は大学の研究室内の 22–25 歳の学部生・大学院生 5 名で，全員男性である．また，全員コンピュータ及びスマートフォンを有しており，日常的に利用している．専門的なデザインの経験はなく，Adobe Illustrator などの DTP ソフトウェアの経験はないが，プレゼンテーション資料などの制作において，簡単なデザイン作業は数度行ったことがある．

まず，実験についての説明を行い,「研究室のポスターを制作する」というタスクを与えた．被験者らが日頃見知っている研究室というテーマを与えることで，デザイン作業の難易度を下げることを狙った．最初にタスクの概要について説明した後，Mac OS X 用アプリケーションである Base について操作方法を説明し，デザイン作業を行わせた．デザインの際に，図 7.19 に示す研究室関連の画像を 1 つ以上，Base の機能では描画できない，研究室のロゴのベクター画像（図 7.18）を 1 つ以上含めることを指示し，その編集には Adobe Illustrator を併用させ，編集後のロゴを Base 側に貼り付けることを指示した．一般的にポスターはいくつか画像を含むものであり，また一般的にロゴなどの編集には Adobe Illustrator などの DTP ソフトウェアを使うため，そして本システムにおける Adobe Illustrator との連携機能を評価するため，このような制約を課した．最後に制作したデザインを送信させた．

次に，iOS 用アプリケーションである Client について操作方法を説明し，自分の制作したポスターの受信を行わせた．次に，研究室の入り口前（図 7.17）において，環境中にすでに存在するポスターの位置に,「貼り付け」インタフェースにより制作した仮想ポスターを貼り付けさせ，白色領域指定による参考色の設定，プレビュー・編集の作業をさせた．その際にはグラフィックの移動，拡大・縮小，パレットによる塗り色の変更，フォントのプレビューという各タスクを一通り体験させた．また，被験者が作業結果に満足した時点で再度 Client から編集後のポスターを送信させ，Base 側で再度受信した結果を確認させた．

最後に，表 7.1 に示すアンケートの各設問への回答を行わせた．回答は 5 段階のリッカート尺度（5:非常にそう思う，4:ややそう思う，3:どちらともいえない，2:ややそう思わない，1:非常にそう思わない）にて回答させた．さらに，自由記述にて感想を述べさせた．

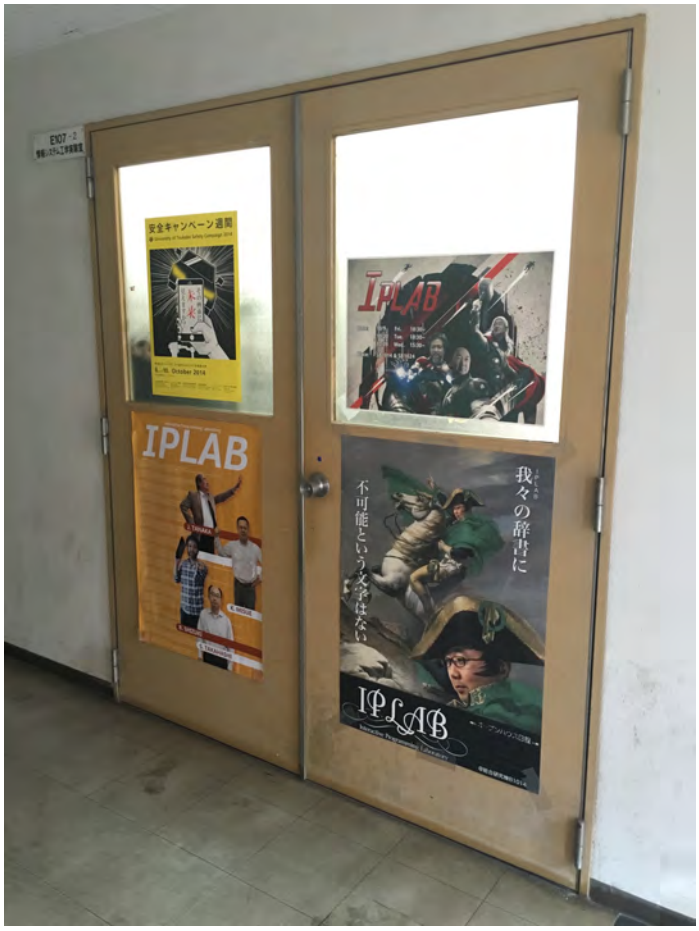

図 7.17: 仮想ポスターを貼る環境

図 7.18: ポスター制作タスクに用いるロゴ

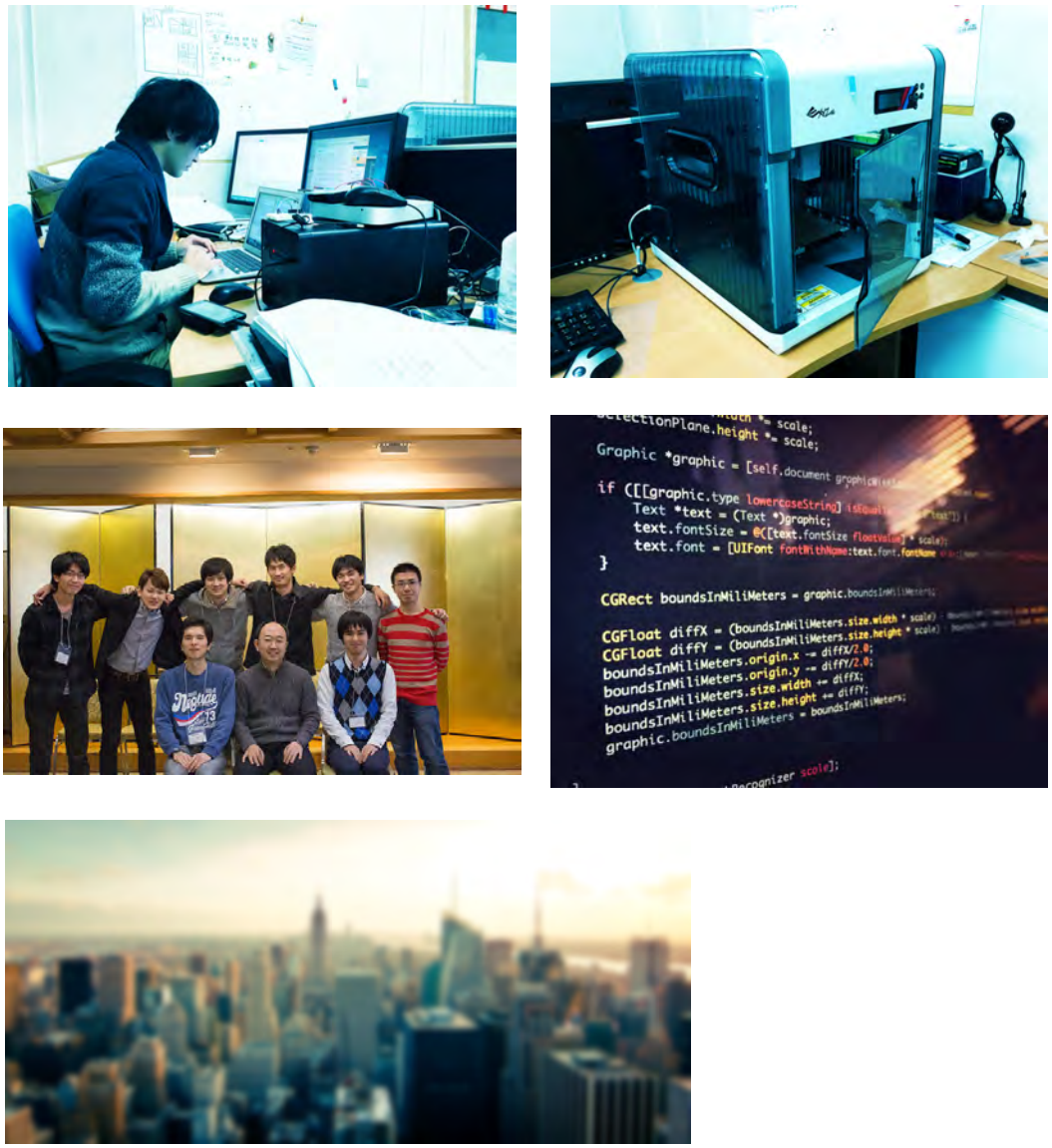

図 7.19: ポスター制作タスクに用いる画像

表 7.1: 実験 2 アンケート

番号	設問
iOS アプリケーションについて	
Q1.	ドキュメントの一覧は見やすかったですか
Q2.	タップによる選択・非選択は使いやすいと感じましたか
Q3.	タッチスクリーンでのドラッグ操作によるグラフィックの移動はしやすかったですか
Q4.	ピンチ操作によるグラフィックの拡大縮小はしやすかったですか
Q5.	パレットによる色変更インタフェースは使いやすいと感じましたか
Q6.	フォントのプレビューは使いやすいと感じましたか
Q7.	実環境を体感しながらのデザイン作業中に，通常の DTP ソフトウェアのみによるデザインに比べて，デザイン上の発見（例：フォントが小さかったか）などがありましたか
Q8.	ポスターを「貼り付け」るインタフェースは使いやすいと感じましたか
Q9.	照明色の指定インタフェースは使いやすいと感じましたか
Q10.	デザインのプレビューは実際の照明色を反映していると感じましたか
Q11.	実際に印刷したポスターを掲示したかのような印象を受けましたか
Mac アプリケーションについて	
Q12.	本アプリケーションでのデザイン制作は，他の類似アプリケーションと同様に行えましたか
Q13.	矩形やテキストなどの挿入は直感的に行えましたか
Q14.	マウスによる画像や矩形などの移動や拡大縮小は違和感なく行えましたか
Q15.	Adobe Illustrator からの貼り付け機能は便利だと感じましたか
自由記述	

7.2.1 結果

5 段階評価によるアンケートの結果を表 7.2 に示す．また，各被験者が Base にて制作したポスター及び，それらを Client にてプレビュー・編集を行った後のポスターを図 7.20〜7.29 に示す．

表 7.2: 実験 2 アンケート結果

番号	設問	非常にそう思う（人）	ややそう思う（人）	どちらともいえない（人）	ややそう思わない（人）	非常にそう思わない（人）	平均値
iOS アプリケーションについて							
Q1.	ドキュメントの一覧は見やすかったですか	2	3	0	0	0	4.4
Q2.	タップによる選択・非選択は使いやすいと感じましたか	0	1	2	2	0	2.8
Q3.	タッチスクリーンでのドラッグ操作によるグラフィックの移動はしやすかったですか	2	2	1	0	0	4.2
Q4.	ピンチ操作によるグラフィックの拡大縮小はしやすかったですか	5	0	0	0	0	5
Q5.	パレットによる色変更インタフェースは使いやすいと感じましたか	1	3	1	0	0	4
Q6.	フォントのプレビューは使いやすいと感じましたか	2	1	2	0	0	4
Q7.	実環境を体感しながらのデザイン作業中に，通常の DTP ソフトウェアのみによるデザインに比べて，デザイン上の発見（例：フォントが小さかったか）などがありましたか	2	1	2	0	0	4
Q8.	ポスターを「貼り付け」るインタフェースは使いやすいと感じましたか	3	2	0	0	0	4.6
Q9.	照明色の指定インタフェースは使いやすいと感じましたか	2	2	1	0	0	4.2
Q10.	デザインのプレビューは実際の照明色を反映していると感じましたか	2	1	1	1	0	3.8
Q11.	実際に印刷したポスターを掲示したかのような印象を受けましたか	1	4	0	0	0	4.2
Mac アプリケーションについて							
Q12.	本アプリケーションによるデザイン制作は，他の類似アプリケーションと同様に行えましたか	0	4	1	0	0	3.8
Q13.	矩形やテキストなどの挿入は直感的に行えましたか	0	3	2	0	0	3.6
Q14.	マウスによる画像や矩形などの移動や拡大縮小は違和感なく行えましたか	2	2	1	0	0	4.2
Q15.	Adobe Illustrator からの貼り付け機能は便利だと感じましたか	4	1	0	0	0	4.8

図 7.20: 実験 2 被験者 A 元デザイン

図 7.21: 実験 2 被験者 A プレビュー・編集後

図 7.22: 実験 2 被験者 B 元デザイン

図 7.23: 実験 2 被験者 B プレビュー・編集後

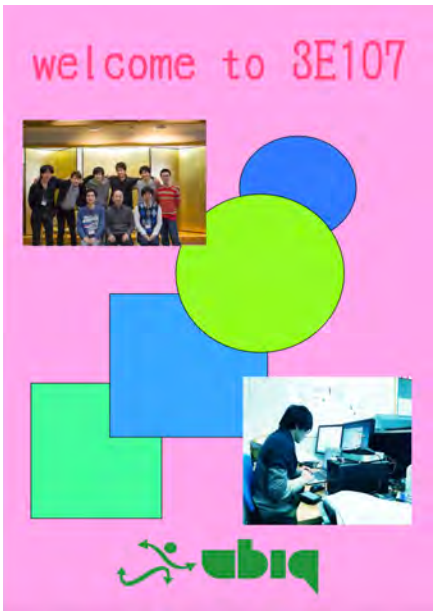

図 7.24: 実験 2 被験者 C 元デザイン

図 7.25: 実験 2 被験者 C プレビュー・編集後

図 7.26: 実験 2 被験者 D 元デザイン

図 7.27: 実験 2 被験者 D プレビュー・編集後

図 7.28: 実験 2 被験者 E 元デザイン

図 7.29: 実験 2 被験者 E プレビュー・編集後

7.2.2　考察

全体として良い評価を得られた．平均値が「ややそう思う」以上となった回答は全回答の 73.3%を占めた．特に，Q3，Q4，Q5，Q6，Q8，Q9 については，提案した DMAR インタフェースの有用性を確認できる結果が得られた．さらに，Q7「実環境を体感しながらのデザイン作業中に，通常の DTP ソフトウェアのみによるデザインに比べてデザイン上の発見などがありましたか」，Q11「実際に印刷したポスターを掲示したかのような印象を受けましたか」で良い評価を得たことから，ポスターのプレビュー・確認を拡張現実で行うという本システムのアプローチの有用性および，このようなシステムの実用性，必要性を確認することができた．自由記述の感想においても，「実際のポスターが他にも並んでいる状況だと他のポスターとの色合いが比べられてよかった」との回答を得られた．また Q10 においては，実験 1 にて確認した照明色の反映結果について，ユーザーも実際に，照明色が反映されていると感じることを確認できた．被験者 A からは「暗めの廊下でポスターを貼り付けた際，実際に少し暗めに表示されていて照明の反映が良く出来ていると感じた」との回答を得た．しかし，別の被験者 D からは「照明を反映しているとはあまり感じなかった」との回答を得た．この原因として，この 2 被験者は実験を行った時刻が異なっており，後者の被験者の方が遅い時間に実験を行ったため，参考色がかなり暗い色となっていたことが考えられる．また，被験者 A の制作したポスターには白色領域が多く，その領域は参考色そのままで描画される一方，被験者 D の制作したポスターは写真を多く含んでおり，そのような領域はほぼ存在しなかった．そのため，「反映された」という印象の受けやすさに差が存在したと考えられる．

一方で，特に低い評価の設問は，Q2「タップによる選択・非選択は使いやすいと感じまし

たか」であり，平均値は 2.8 であった．タップによる選択・非選択については，自由記述の感想においても「タップで選択・非選択をするのが少しわずらわしいと感じた」「タップの選択で，別のものをタップするとフォーカスが変わるのではなく複数選択となるのは違和感を感じた」「タップによる選択／非選択切り替えよりも，タップで 1 つ選択，ロングタップのときは追加選択，非選択の方が良いのではと思った」という評価であった．以上より，選択タスクについては，より良いインタラクションメソッドを模索するべきである．

さらに，アプリケーション改良案として「iOS 側で元の画像と編集後の画像が 1 操作で見比べられればと思った」というコメントを得た．確かに，本システムで実環境でのプレビューは可能であるが，元のデザインと比べてよくなったかを切り替えて表示できれば，さらにプレビューの利便性は向上すると考えられる．このような機能を有するソフトウェアの例として，写真編集ソフトウェアの Adobe Photoshop が挙げられる．Adobe Photoshop では，フィルタの適用前と適用後を即座に切り替えて，効果の程を確認する機能が備わっている．

以上より，全体的な有用性については確認できたものの，個々のインタラクションメソッドや機能については課題があることが明らかとなった．今後は，まずは選択タスクにおけるインタラクションメソッドについて，様々な手法を試すことが必要である．特に，得られた回答に共通している，あるグラフィックを選択した状態で，他の部分をタップするとフォーカスが外れる／選択解除となるのではなく，複数選択となってしまうのがわかりづらい，という部分の修正を行いたい．実験中に，編集を行うユーザーを観察したが，複数のグラフィックをまとめて操作する，というケースは少数であった．これらの回答及び観察結果から，複数選択に必要な操作の手数と引き換えにでも，単一のグラフィックの選択というメジャーなタスクにおけるインタラクションメソッドの使い勝手を向上させるべきであることがわかった．

第8章　結論

本研究では，ポスター等の印刷物のグラフィックデザイン制作において，コンピュータのみで行うDTP作業の限界と問題点を指摘し，実環境におけるグラフィックデザインのプレビュー及び編集を，拡張現実技術を用いて可能にした．まず，グラフィックデザインのプレビュー及び編集を拡張現実を用いて行う際に，ユーザーが達成する必要のあるタスクについて考察し，4つのタスクを提案した．さらに，実環境にすでに掲示されているポスターの，フォントの編集を可能にするプロトタイプシステムを開発し，拡張現実によるプレビュー・編集の有用性を確認，各タスクに対するインタラクションメソッドを提案した．さらに，プロトタイプシステムの予備実験から得られた知見を元に，より現実の業務に近いデザインフローにも対応させるべく，仮想ポスターによるデザインシステムを提案した．主となるデザイン制作を行うデスクトップコンピュータ用アプリケーションとARにより仮想ポスターをプレビュー・編集できるスマートフォンアプリケーションを連携させることにより，コンピュータによって制作したデザインを即座に現場にてプレビュー・編集することを可能とした．また，仮想物体を現実に重畳表示し現実に存在しているかのように見せる際に問題となる，光学的整合性の問題に対して，白色領域を指定しその色を仮想物体に乗算する手法を提案し，実環境に印刷物を掲示した際の見え方に近いプレビューを達成した．さらに，前述のARおよびインタラクションメソッドを含んだモバイル端末およびコンピュータからなるシステムについて，各種機能と実装方法について詳細に示した．最後に，システムの評価実験においては，提案した照明色反映手法の有用性を定量的に示し，アンケート調査にて，仮想ポスターによるデザインシステムがグラフィックデザイン制作において有用であることを示した．

今後の発展として，アンケート調査から明らかになった，グラフィック選択手法の問題点の改善が挙げられる．

謝辞

本論文を執筆するにあたり，指導教員である高橋伸准教授には懇切丁寧なご指導を賜りました．あらためまして，ここに厚く御礼申し上げます．また，田中二郎教授、三末和男教授、志築文太郎准教授、嵯峨智准教授、Simona Vasilache 助教にも、全体ゼミでの発表練習などを通して大変貴重なご意見・ご協力を頂きました．心より御礼申し上げます．

またインタラクティブプログラミング研究室の皆様には，日頃の研究室での生活を通して多くのアドバイスや示唆に富むコメントをいただきました．特にユビキタスチームの皆様とは大学院生活の 2 年間を通じて共に研究活動を行い，お互いに切磋琢磨しあえたかと存じます．あらためまして感謝の意を表します．

最後に，ここまで私を支えてくださいました友人や家族をはじめとする，お世話になったすべての方々に心より感謝いたします．

参考文献

[1] 蔵田武志, 清川清, 大隈隆史. AR（拡張現実）技術の基礎・発展・実践. 科学情報出版, September 2015.

[2] J. Grudin. Computer-supported cooperative work: history and focus. *Computer*, Vol. 27, No. 5, pp. 19–26, May 1994.

[3] Tom Rodden. A survey of cscw systems. *Interacting with Computers*, Vol. 3, pp. 319–353, 1992.

[4] Klaus H Ahlers, André Kramer, David E Breen, Pierre-Yves Chevalier, Chris Crampton, Eric Rose, Mihran Tuceryan, Ross T Whitaker, and Douglas Greer. Distributed augmented reality for collaborative design applications. In *Proceedings of the Computer Graphics Forum*, Vol. 14, pp. 3–14. Wiley Online Library, 1995.

[5] Gudrun Klinker, Allen H. Dutoit, M. Bauer, J. Bayer, V. Novak, and D. Matzke. Fata Morgana - a presentation system for product design. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pp. 76–85, 2002.

[6] Woohun Lee and Jun Park. Augmented Foam: a tangible augmented reality for product design. In *Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 106–109, Oct 2005.

[7] Kiyoshi Kiyokawa, Haruo Takemura, and Naokazu Yokoya. Seamlessdesign for 3d object creation. *IEEE multimedia*, No. 1, pp. 22–33, 2000.

[8] Seungwon Kim, Gun A. Lee, Sangtae Ha, Nobuchika Sakata, and Mark Billinghurst. Automatically freezing live video for annotation during remote collaboration. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pp. 1669–1674, New York, NY, USA, 2015. ACM.

[9] Michael Rohs. Real-world interaction with camera phones. In *Ubiquitous Computing Systems*, pp. 74–89. Springer, 2005.

[10] Johannes Schöning, Antonio Krüger, and Hans Jörg Müller. *Interaction of mobile camera devices with physical maps*. 2006.

[11] Wolfgang Hrst and Casper van Wezel. Gesture-based interaction via finger tracking for mobile augmented reality. *Multimedia Tools and Applications*, Vol. 62, No. 1, pp. 233–258, 2013.

[12] Shahrouz Yousefi, Farid Abedan Kondori, and Haibo Li. Experiencing real 3d gestural interaction with mobile devices. *Pattern Recognition Letters*, Vol. 34, No. 8, pp. 912 – 921, 2013. Computer Analysis of Images and Patterns.

[13] Shunichi Kasahara, Valentin Heun, Austin S. Lee, and Hiroshi Ishii. Second surface: Multi-user spatial collaboration system based on augmented reality. In *SIGGRAPH Asia 2012 Emerging Technologies*, SA '12, pp. 20:1–20:4, New York, NY, USA, 2012. ACM.

[14] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. Drillsample: Precise selection in dense handheld augmented reality environments. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*, VRIC '13, pp. 10:1–10:10, New York, NY, USA, 2013. ACM.

[15] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. 3DTouch and HOMER-S: Intuitive manipulation techniques for one-handed handheld augmented reality. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*, VRIC '13, pp. 12:1–12:10, New York, NY, USA, 2013. ACM.

[16] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. I–355–I–362 vol.1, 2001.

[17] Georg Nebehay and Roman Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2784–2791, 2015.

[18] Tetsuya Kakuta, Takeshi Oishi, and Katsushi Ikeuchi. Virtual kawaradera: Fast shadow texture for augmented reality. *Proceedings of Intl. Society on Virtual Systems and MultiMedia*, pp. 141–150, 2004.

[19] Masayuki Kanbara and Naokazu Yokoya. Geometric and photometric registration for real-time augmented reality. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, pp. 279–, Washington, DC, USA, 2002. IEEE Computer Society.

[20] M. Kanbara and N. Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *Proceedings of the 17th International Conference on Pattern Recognition*, Vol. 2, pp. 911–914 Vol.2, Aug 2004.

[21] About the Cocoa document architecture, Apple Mac Developer Library. https://developer.apple.com/library/mac/documentation/DataManagement/Conceptual/DocBasedAppProgrammingGuideForOSX/Introduction/Introduction.html.

[22] Archives and serializations programming guide, Apple Mac Developer Library. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Archiving/Archiving.htm.

[23] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pp. 85–94, 1999.

[24] Scenekit + vuforia で AR アプリを作ろう - Qiita. http://qiita.com/akira108/items/e5b43810b64cd921a947.

[25] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 22, No. 11, pp. 1330–1334, Nov 2000.

[26] 藤本雄一郎, 青砥隆仁, 浦西友樹, 大倉史生, 小枝正直, 中島悠太, 山本豪志朗. OpenCV3 プログラミングブック. マイナビ出版, September 2015.