Vol.2021-DPS-189 No.7
2021/12/20

インターネット接続に依存しない Location-based AR基盤の一考察

大畑 誠弥[1,a)]　戸河 圭太[1]　山本 隼矢[1]　片瀬 拓海[1]　中沢 実[1]

概要：xR 技術は人間を視覚的に拡張する手段の一つである．その中の一つに Location-based AR が挙げられる．これは GPS や BLE ビーコンなどを利用して，クライアントのリアルな位置情報と AR オブジェクトとを紐づけて描画する技術である．しかし，描画するオブジェクトがクライアント内部になくインターネットを経由して取得する場合，インターネット接続が必須である．また，多くの Location-based AR では位置情報とデバイスが向いている方位などを利用して描画するため，リアルな位置情報とデバイスが取得した情報の間において誤差がたびたび発生し，描画がズレてしまう場合がある．本提案では，AR サーバを対応した空間それぞれに配置してインターネット接続への依存を無くし，Marker-based AR を採用した描画のズレの少ない Location-based AR を実現する．

キーワード：AR，xR，Location-based AR，Marker-based AR

Study of Location-based AR infrastructure that does not depend on internet connection

1. はじめに

xR 技術は人間を視覚的に拡張する手段の一つであり，仮想現実（VR, Virtual Reality）や拡張現実（AR, Augmented Reality），複合現実（MR, Mixed Reality）の総称である．現代において，xR 技術はゲームやライブ，各種作業現場のナビゲーション機能など，幅広く活用されている．その中でも AR はスマートフォンやタブレットの基本的な OS である iOS や AndroidOS 向けの開発用フレームワークが提供されており，手軽に高機能な AR アプリケーションの開発・提供が可能になっている．AR は目の前の平面に任意のオブジェクトを描画したりマーカー画像に紐づいて描画（Marker-based AR）したり，ときには描画空間を共有するものなど様々である．その中の一つに，位置情報に基づいて AR オブジェクトを描画する Location-based AR がある．

Location-based AR は AR デバイスの実際の位置情報に基づいて AR オブジェクトを描画する．また，AR デバイスの位置情報のみならず，各種センサによって得られたデバイスの方位や加速度を利用する場合もある．一般的に Location-based AR で描画するオブジェクトは，アプリケーションにプリインストールされているものや起動時など任意のタイミングでインターネット上のサーバから取得するものがある．

しかし，インターネット上のサーバから取得する場合，オブジェクトの取得がインターネット接続に依存するため，インターネット接続が切れた時やインターネットへの接続が困難な環境下では，AR オブジェクトの描画ができなくなる．Location-based AR は実空間や実座標に紐づいたコンテンツにも関わらず，その場に居てもインターネット接続がなければコンテンツ提供できない現状は不都合である．

そこで本稿では，3D オブジェクトをあらかじめ保持せず，尚且つインターネット接続に依存しないでその空間で完結する Location-based AR システムを提案する．インターネット接続に依存しないために，オブジェクトを描画したい実空間にサーバとなるコンピュータを配置し，マーカーとオブジェクトをセットで送信することで，Marker-based

[1] 金沢工業大学
Kanazawa Institute of Technology
[a)] b1802034@planet.kanazawa-it.ac.jp

AR な Location-based AR を実現する．また，GPS などの位置情報ではなく，Marker-based AR を採用し描画精度の向上を図る．本システムは，特定の場所や部屋全体への AR オブジェクトの描画を想定している．描画する AR オブジェクトはいくつあってもよく，現実空間の装飾などの拡張に使用できる．

2 章では AR 分野の近年の動向について紹介する．3 章では本システムの概要を説明し，4 章では詳細な実装や仕様を紹介する．5 章では本システムがどのように拡張可能かについて考察する．

2. 動向調査

近年 Location-based AR を含む AR 技術は盛んに応用されている．

河村，大槻らの放送番組と同期する自由視点 AR のためのリアルタイム伝送システム [2] では，テレビの放送に合わせて AR オブジェクトを描画している．AR オブジェクトは放送されるテレビのコンテンツに合わせたものになっており，スマートフォンやタブレットを AR デバイスとして目の前の平面へ描画している．3D オブジェクトのメッシュやテクスチャなどの静的データはあらかじめダウンロードし，モーションファイルをリアルタイムに受け取る．それらを AR デバイス内で結びつけて描画することで，テレビの放送に同期したリアルタイムなコンテンツの提供を可能にしている．

AR を行う場合，大きく分けてマーカーを利用する場合と利用しない場合がある．マーカーを利用する場合は，認識したマーカー画像に合わせて AR オブジェクトを描画する．利用しない場合，デバイス座標との相対で描画するか GPS を用いて位置情報に基づいて描画する，認識した平面などに合わせて描画するなどの方法がある．しかしこれらのマーカーを利用しない場合の方法では，多少の誤差によってズレが生じたり，描画範囲に制約があったり定まらなかったりする上に，修正も難しい．他にも Apple 社の ARWorldMap[6] のように認識した特徴点情報用いて記憶し，同一空間を判定する方法も存在する．しかし，これでは同一空間なのにも関わらず，物が移動しただけで別空間だと認識してしまう可能性が高い．

Jack, Keyu らの Comparison of marker-based AR and markerless AR: A case study on indoor decoration system[1] では，AR にマーカーを利用した場合と利用しない場合の比較を行なっている．ここで言う markerless な AR は主に位置情報に GPS を使った場合である．Marker-based AR はマーカー画像の複雑さによって安定感が増すが，明るさなどによっては安定しないと述べている．markerless な AR の場合，室内 GPS の値が急激に変化しうるため，他の要素と複合で描画を安定させる必要がある．精度に関しては Marker-based AR は問題にならないほど良いが，markerless な AR の場合は細かい調整は難しい．GPS の精度は 10m 未満であり，特定の部屋などにリンクすることは不可能である．

高橋，佐藤らの複数の屋内測位方式を用いた電動車椅子自動運転システムの提案 [3] では，位置測位手法の一つに Marker-based AR の技術を転用している．Marker-based AR は本来実空間上のマーカー画像を認識し，その上に AR オブジェクトを描画する技術である [4]．この研究では AR オブジェクト描画はせず，座標計算の部分のみ利用している．Marker-based AR は Marker-less に比べて描画精度が良いことが知られているが，このようにその部分のみ転用しているものも存在する．

佐々木，平川らのビーコンを利用した拡張現実による観光情報提供システム [5] では，BLE ビーコンを用いた Location-based AR を実現している．BLE ビーコンの ID によって場所を識別し，インターネット上のサーバと通信してコンテンツを取得したのち描画している．コンテンツは，AR デバイスの正面にその場所に関連した画像が表示される．

3. 提案システム

本稿では，3D オブジェクトをあらかじめ保持せず，尚且つインターネット接続に依存しないでその空間で完結する Location-based AR システムを提案する．

3.1 システム概要

提案システムは，実空間を空間ドメインによって分離し，各空間ドメインごとに AR オブジェクトの描画に必要な情報（以降，AR 描画データ）を保持・配布するビーコン（以降，AR ビーコン）を一つ配置する．AR 描画データは配置された空間ドメインに紐づいており，その空間ドメインに所属するクライアントに配布される．クライアントは，自身が所属する空間ドメインの中から AR ビーコンを検索・通信し，受け取った AR 描画データを元に AR オブジェクト群を描画する．

AR ビーコンと空間ドメインは N:N で対応する．つまり一つの AR ビーコンが複数の空間ドメインに参加してもよく，複数の AR ビーコンが一つの空間ドメインに参加してもよい．複数の AR ビーコンが空間ドメイン内にある場合，クライアントは全ての AR ビーコンから AR 描画データを取得する．AR ビーコンとクライアントの関係を**図 1** に示す．

AR 描画データは，描画の基点となるマーカー画像ファイルと，それに紐づく空間に対応した AR オブジェクトのファイルで構成される．一つのマーカー画像に対して，一つ以上の AR オブジェクトのファイルが対応し，このセットはいくつあってもよい．このマーカー画像を用いて空間ドメインごとに Marker-based AR を実行する．

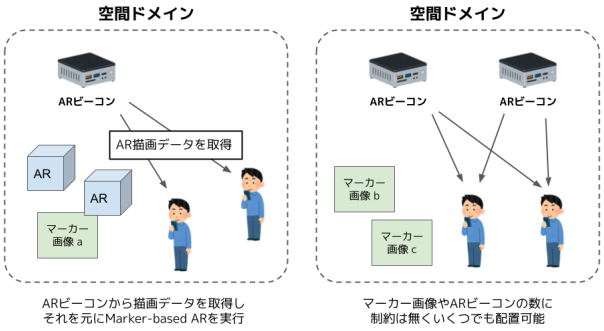

図 1 システムの全体像

また，AR ビーコンは周囲の AR ビーコンに AR 描画データをバックアップできる．定期的に周囲の AR ビーコンを探索し，見つかれば自身が持っている AR 描画データを他の AR ビーコンに送信する．それにより，AR ビーコン起動時に周囲の AR ビーコンが持つ自分のバックアップから，AR 描画データを復元できるようになる．

このシステムはその場所ごとの AR オブジェクトをインターネットに依存せず取得・描画できる．部屋の装飾をしたり看板などを置いたり，その場所に紐づいた様々な使い方が想定される．

3.2 空間ドメインの分離方法

空間ドメインはネットワークベース・距離ベースの二種類の方法で分離できる．空間ドメインの分離方法を**図 2** に示す．

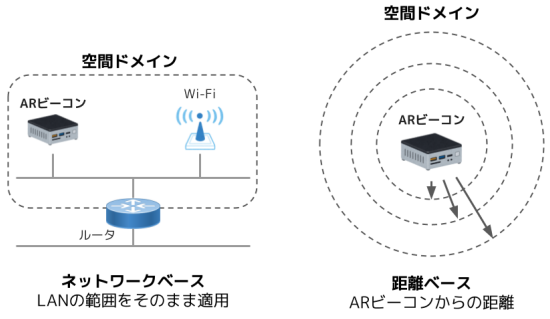

図 2 空間ドメインの分離方法

ネットワークベースの空間ドメイン分離は，空間ドメインの範囲に主に LAN（Local Area Network）を用いる．この場合，LAN そのものが空間ドメインと 1:1 で対応する．AR ビーコンが接続されている LAN が，その AR ビーコンが管轄する空間ドメインとなる．この場合，空間ドメインの広さは LAN の広さに直結するため，空間ドメインの制御は LAN の制御によって決まる．

距離ベースの空間ドメイン分離は，空間ドメインの範囲に距離を用いる．実際には AR ビーコン自体が発する Wi-Fi の電波強度が閾値以上なら，そのドメイン内にいると判定する．Wi-Fi の電波強度は周りに何もなければ距離で減衰するため，一定距離の範囲を表現できる．

3.3 描画方式について

描画には Marker-based AR を採用した．Marker-based AR は AR オブジェクトの描画方式の一つで，特定の画像をマーカーとし，そのマーカー画像を認識したら対応したオブジェクトをマーカー画像を基点に描画・追従する．マーカーとして使う画像は任意のものが使用できる．実空間上の画像認識ベースで行うため描画時のズレが少なく，描画座標の修正もマーカー画像を再度見るだけで行える．

空間ドメイン内に存在する特定の画像を登録することで実現でき，単に AR ビーコンを配置すれば良い今回の提案と親和性が高い．実際には，AR 描画データに含まれる一つのマーカー画像を基点に，対応した複数の AR オブジェクトを描画する．

4. 実装

この提案システムの実現方法や実装について，サーバサイドとクライアントサイドに分けて説明する．また，実装したシステムの速度についても検証する．

4.1 実装概要

このシステムはサーバサイドとクライアントサイドに分かれる．サーバサイドには，所属する空間に紐づいた AR 描画データを保持・配布する AR ビーコンが相当する．クライアントサイドはその空間にある AR ビーコンを探し，見つけたら AR 描画データを受け取り描画する．それぞれの GitHub リポジトリ [7][8] として公開している．今回の実装ではサーバサイドに Raspberry Pi4 ModelB を，クライアントサイドに iPhone 12Pro を使用する．AR 描画データのやりとりを**図 3** に示す．

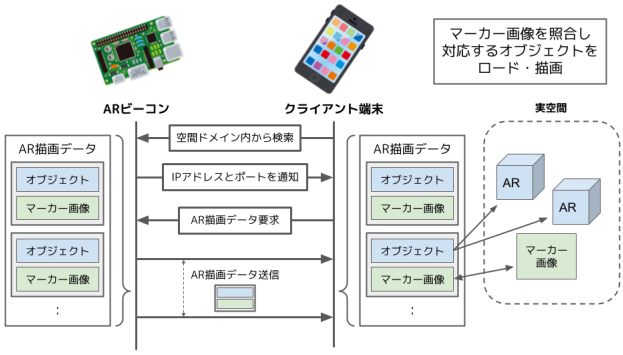

図 3 AR 描画データの扱い

AR 描画データをやりとりするプロトコルは TCP,UDP を使用する．UDP は空間ドメイン内で AR ビーコンを探すブロードキャストに使用し，それ以外の通信は TCP で

行う．TCP 上でデータをやるとりする上で，単純なバイト列だと細かいデータのパースや汎用性の面で不便なため，新しく構造体 LLDARSLayer を定義した．それを以下に示す．

```
// LLDARSLayer構造体（golang）
type LLDARSLayer struct {
    Type        LLDARSLayerType
    ServerId    uint32
    Origin      uint32
    ServicePort uint16
    Length      uint64
    Payload     []byte
}
```

上から順番に，メッセージタイプ，サーバの UUID，送信元 IP，ポート情報，ペイロードの長さ，ペイロード本体となる．LLDARSLayerType はメッセージの種類を表すフィールドであり，uint8 の列挙型で定義される．Service-Port フィールドを使うポート番号は，内部的に 3 つ定義されている．実際にサービスを待ち受けるポートとサーバが他のサーバを探索するときに使うポート，クライアントがサーバを探すポートがある．サーバを探すポートが 2 つに分かれているのは，空間ドメインをネットワークで区切るとき，どちらかのブロードキャストだけをフォワードできるようにするためである．

サーバサイドの実装は Go 言語を用い，Linux 上で動くことを想定し実装する．クライアントサイドの実装は Apple 社の ARKit[9]/RealityKit[10] を用いる．ARKit や RealityKit は手軽に iOS で AR 機能を実装できるフレームワークである．描画する AR オブジェクトのデータは，RealityComposer[11] で作成できる USD ファイル（usdz）を使用する．Marker-based AR で描画する時のマーカー座標が，USD ファイルの原点座標と対応する．USD ファイルは内部に 1 つ以上のオブジェクトを内包でき，1 つ以上の USD ファイルを 1 つのマーカー画像に紐づけて描画する．また，これらの AR 描画データはサーバには ZIP 形式で保存され，クライアント側で伸長して使う．

4.2 サーバサイド

サーバサイドは AR ビーコンとして実装する．AR ビーコンは UUID で全て一意に識別する．AR ビーコンでは以下の 5 つを並列に実行する．

（1） 送信や同期などの待受

（2） 周囲の AR ビーコンに対してバックアップ要求

（3） クライアントからのブロードキャスト待受

（4） 他 AR ビーコンからのブロードキャスト待受

（5） Wi-Fi のアクセスポイントとして機能

(1) では，メッセージタイプからどのサービスが要求されているかを判断し実行する．ここで受け取りうるメッセージの種類は，クライアントからの AR 描画データの要求・他サーバからのバックアップ要求・他サーバからの復帰要求の 3 つである．一つ目のクライアントからの AR 描画データの要求の場合は，Origin の IP アドレスに対して TCP で AR 描画データを送信する．二つ目の他サーバからのバックアップ要求の場合は，他サーバから送られてくるそのサーバがホストする AR 描画データを受信し保存する，三つ目の他サーバからの復帰要求の場合は，事前にバックアップとして受け取っていたそのサーバの AR 描画データを送信する．
(2) では，周囲のサーバを定期的に探索し，発見した場合にバックアップ要求を送る．探索にはクライアントと同じくネットワークベースと距離ベースの方法を使用する．
(3)(4) では，共にブロードキャストの待受をする．ここで異なるのは待受ポートだけである．ブロードキャストを受け取ったら，送信元に対して自身の IP アドレスと (1) を待ち受けるポートを通知する．
(5) では，距離ベースの通信を行うために特定の文字列と AR ビーコンの UUID で構成される SSID の Wi-Fi を発する．この Wi-Fi ネットワークでは，動的に IP アドレスが割り当てられる．

4.3 クライアントサイド

クライアントサイドは iOS のアプリケーションとして実装する．このアプリケーションの基本部分は Marker-based AR アプリケーションである．自身が所属する空間ドメインに配置されている AR ビーコンから，AR 描画データであるマーカー画像と USD ファイルを受け取ることで実現させる．

このアプリケーションは，以下の流れで実行される．

（1） Wi-Fi に接続されているか確認

（2） 接続されていなければ，距離ベースで周囲の AR ビーコンから発される Wi-Fi を検索し接続

（3） 所属している Wi-Fi で AR ビーコン探索（ブロードキャスト）

（4） AR ビーコンが見つかれば，AR 描画データを要求

（5） AR 描画データを受信

（6） 受信終了次第，Marker-based AR を実行

AR ビーコン探索の際，距離ベースの場合順次 Wi-Fi に接続して AR 描画データを取得していく．これは Wi-Fi には同時に一つしか接続できないからである．ネットワーク

ベースの場合は，一度のブロードキャストで所属する空間ドメインにある AR ビーコンが全て列挙できる．そのため距離ベースでの探索よりも早い．

Marker-based AR を実行する際，AR 描画データは ZIP から伸長しマーカー画像と USD ファイルのパスとを紐づけて記憶する．AR オブジェクトとなる USD ファイルは，対応するマーカー画像が認識されたタイミングで初めて読み込まれる．これはその場に関係ないものや今すぐには必要ないオブジェクトが大量に送られた際に，メモリと処理速度を犠牲にしないためである．AR オブジェクトの描画はマーカーに追従し，画面から外れたらその場に保持するように実装する．よって，常にマーカー画像を画面に収める必要がなく，また画面内に収まりきらない大きさや座標の AR オブジェクト群も扱える．

4.4 配送・描画速度

このシステムの実用性を検証するため，AR ビーコンからの描画データ取得時間と，初回マーカー認識時の描画時間を計測した．検証に使うデバイスは，前述した AR ビーコンとクライアント端末と同一のものである．AR 描画データは ZIP 形式のファイルで，中身のオブジェクトファイルは RealityComposer から書き出せる USDZ ファイル，マーカー画像ファイルは 607kB の JPEG ファイルを使用する．USDZ ファイルは USD ファイルを ZIP 圧縮したものであり，通常の USD ファイルよりもファイルサイズが小さい．ネットワーク環境は，AR ビーコンが有線でクライアントデバイスが無線で同一ネットワークに所属する状態を想定する．帯域は 100MB/s で，iperf3[12] で計測した実効速度は 45MB/s ほどだった．

検証には**表 1** のファイルサイズが異なる 5 つのデータを使用した．計測は，AR ビーコンにあらかじめ各データを保存し起動した状態で，クライアントアプリ側の実行時に行う．それぞれのデータに対する計測は独立しており，キャッシュなどが効かないよう都度アプリケーションの削除・再インストールを行う．

表 1 検証に使用するデータ

データ名	AR 描画データ	USDZ ファイル
a	622 KB	1 KB
b	2.0 MB	1.5 MB
c	11.5 MB	11 MB
d	45.4 MB	47 MB
e	87.7 MB	99 MB

結果を**表 2** に示す．

計測結果について，取得時間については概ねネットワークの実効速度の通りに推移している．描画時間については，USDZ ファイルの大きさとの関係が取得時間ほどは見られない．これは USDZ ファイルが USD ファイルのアーカイブ形式であり，ファイルサイズが実際の複雑度とは乖離している可能性があるからだと推測できる．USDZ 形式では，同一オブジェクトを複数個複雑な座標に配置したとしても，そのオブジェクト単体のデータ量と大きく違わないファイルサイズとなる．そのため，描画時間の評価に関しては USDZ ファイルの大きさとの単純比較だけでは十分ではなく，実際にオブジェクトを描画する際の複雑度や物量などの情報量といった指標も必要である．

表 2 計測結果

データ名	取得時間 [ms]	描画時間 [ms]
a	350	400
b	530	1140
c	1610	1440
d	4740	3610
e	9400	4920

5. 今後の課題と展望

計測結果に関して，アプリケーションの見せ方次第では問題にならない速度だと考えられるが，さらに大きい AR 描画データや複雑なオブジェクトを扱う場合に問題になる可能性が高い．今回は実装と取り回しの容易さゆえに AR 描画データをひとまとめに圧縮したが，AR 描画データの取得から描画までの時間を短縮するために，オブジェクトファイル一つ一つをバラバラに扱い順次描画するなどの工夫が必要である．また，距離ベースでのフローにおいて，最初から Wi-Fi の SSID を検索するのではなく，事前に BLE などで AR ビーコンが発する Wi-Fi の SSID やパスワード，AR ビーコンの IP アドレスなどをやりとりすることで，動的な SSID やパスワードを使用できるとともに Wi-Fi 接続後の AR ビーコンの検索処理を省ける．

今回使用したのは単純な AR オブジェクトだが，アニメーションのあるものや，壁や床などの平面を認識して描画するものなど，様々な使い方が考えられる．と同時に，手軽に AR ビーコンにオブジェクトファイルを追加・編集できるために，現実空間に合わせてオブジェクトを作るツールの拡充も必要である．また，これら AR ビーコンを木構造を持たせられれば，配布する AR 描画データをうまく構造化できると考える．

6. まとめ

本稿では，インターネット接続に依存しない Location-based AR 基盤として，AR ビーコンを用いた Marker-based な Location-based AR を提案し，Go・ARKit を用いた実装を報告した．実空間に配置されるコンピュータから，描画の基点となるマーカー画像と実際に描画するオブジェクトファイルをクライアント端末に配布することで実現した．今後は，描画までの時間を短縮するとともに，よりユーザ側に寄ったツールの検討を進める．

参考文献

[1] Jack C.P. Cheng, Keyu Chen, Weiwei Chen
Comparison of marker-based AR and markerless AR: A case study on indoor decoration system
Lean & Computing in Construction Congress(LC3)(2017)
[2] 河村侑輝，大槻一博
放送番組と同期する自由視点 AR のためのリアルタイム伝送システム
AVM-103 No.11(2018.11)
[3] 高橋宗資，佐藤文明
複数の屋内測位方式を用いた電動車椅子自動運転システムの提案
第 26 回マルチメディア通信と分散処理ワークショップ論文集 (H30.11)
[4] Anuroop Katlyar, Karan Kalra, Chetan Garg
Marker Based Augumented Reality
Advances in Computer Science and Information Technology(ACSIT)(2015)
[5] 佐々木克海，平川剛，橋本浩二，柴田義孝
ビーコンを利用した拡張現実による観光情報提供システム
情報処理学会全国大会公演論文集 (2016)
[6] ARWorldMap — Apple Developer Documentation
https://developer.apple.com/documentation/arkit/arworldmap
[7] silmin/LLDARS: Location-based Local Distributed AR System
入手先 ⟨https://github.com/silmin/LLDARS⟩
[8] silmin/LLDARS-ios-client: LLDARS Client iOS App
入手先 ⟨https://github.com/silmin/LLDARS-ios-client⟩
[9] ARKit Overview - Augmented Reality - Apple Developer
https://developer.apple.com/augmented-reality/arkit/
[10] RealityKit 2 Overview - Augmented Reality - Apple Developer
https://developer.apple.com/augmented-reality/realitykit/
[11] AR Tools - Augmented Reality - Apple Developer
https://developer.apple.com/augmented-reality/tools/
[12] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool
https://iperf.fr/