コンピュータ理工学特別研究報告書

題目

TrueDepth カメラを用いた FaceTracking による アニメ風 3 次元 CG モデルの表情操作

学生証番号　744311

氏名　小田原 史弥

提出日　令和 3 年 1 月 22 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

ARKit は Apple 社が開発した iOS デバイスのためのフレームワークである．ARKit を用いることで,AR アプリの開発が容易になる.FaceTracking は ARKit の機能の一つで，iPad Pro などに装備されている TrueDepth カメラを用いて，顔の追跡や表情認識を行うことができる．本研究では，FaceTracking で認識した顔の表情を自作の顔の 3D モデルに投影することで，自分の表情を自分の 3D モデルで表現できる技術を開発した.

自作の顔の 3D モデルは,Blender で制作した.髪の部分と顔の部分に分けて制作し，それを合成した．Blender では，顔の表情が表現できるように，まぶたや眼球など顔パーツの変形をシェープキーと呼ぶ技術で実装する．ARKit の FaceTracking で，これらの顔パーツを制御できるようにシェープキーを作成した．完成した自作の顔 3D モデルを Collada 形式でファイル保存し，開発した swift のプログラムで読みこむ.

ARKit を使った swift のプログラムの実行中，FaceTracking の結果が ARFaceAnchor オブジェクトのプロパティに保存される．この中で，顔の表情は blendShapes という名前付き係数辞書に保存される．これは，eyeBlinkLeft など目・鼻・口の 52 個の特徴点の位置を 0 から 1 の値で保存した表情を表現するデータである.開発したプログラムでは，入力した画像に FaceTracking を実行するごとに，blendShapes の値で 3D モデルの形状を制御した（swift のプログラムでは morpher に値をセットする）．動作確認を行い，喜怒哀楽の中の喜びと怒りの表情を 3D モデルと実際の表情を比較した結果， 3D モデルの顔の表情に正しい変化を与えることができ,表情を大きく変化させた時のモデルの表情と本来設定した表情が一致した事から動作が正常であることが確認できた.

目次

1章 序論

AR（Augmented Reality，拡張現実）技術は，実写の風景映像にバーチャルな視覚情報を重ねて表示することで，実世界を仮想的に拡張するような技術である[1]．最近，スマホやタブレット端末のアプリとして， AR 技術を用いたものが多数リリースされている．有名なものに，ゲームアプリである「Pokèmon Go」やカメラアプリである「SNOW」などがあり，一般ユーザは知らないうちに AR に触れているケースが少なくない．

図 1 の「Pokèmon Go」の例では，通常のカメラで撮影した写真画像に，３次元 CG による仮想オブジェクトを置いたように表示している．また図 2 の「SNOW」の画面では，手に火が灯っているように撮影されている．

図 1　「Pokèmon Go」の画面(左)と普通のカメラで撮影した写真(右)

図 2 「SNOW」で撮影した手の写真

AR 技術の特徴は，CG 画像と実写画像を単に重畳するだけではなく，現実世界から得た情報を使って，重畳する CG 画像を制御・調整することで，違和感のない画像合成を実現していることである．現実世界から得る情報として，「現在位置」と「周囲環境」に関する情報が特に重要である．GPS などの位置情報に紐付けて情報を表示する AR をロケーションベース型と呼び，カメラ画像を使った物体認識や空間認識を使用する AR をビジョンベース型と呼ぶ．本研究の AR はビジョンベース型 AR である[2]．

ビジョンベース型の AR は画像認識型とも言われている．コンピュータビジョンによる画像認識や空間認識を利用して周囲環境を解析し，その結果に基づいて情報提示を行う事が特徴である．この時，処理を容易にするために，特別な形状をした AR マーカーを被写体に貼り付け，それを端末のカメラに認識させるものをマーカー型と呼ぶ．特定の AR マーカーを用いずに，実空間に存在する物体や建造物，山などの情景をあるがままに認識し，情報を表示させるものをマーカーレス型と呼ぶ．本研究で扱う AR 技術は

素顔から顔パーツの動き認識し，その結果に基づいて情報を表示するので，マーカーレス型にあたる[2].

本研究で使用するTrueDepthカメラは，iPad，iPhoneに備え付けられている特別なカメラであり，iPadはiPad Pro（CPUがA12Xの製品），iPhoneはiPhone X以上に備わっている．このTrueDepthカメラは，3万以上の目に見えない赤外線によるドットを被写体に照射して撮影することで，被写体の奥行き情報を得ることができる．顔を撮影する場合，顔の深度マップを作成し，顔の正確な形状データを読み取る事ができる[3]．このようにして得たデータを，Appleが提供するフレームワークの一つであるARKitで利用できる．

ARKitはiOSで利用できるフレームワークの一つで，これを使うことでARアプリの開発が容易になる．ARKitはiOSデバイスのフロントカメラまたはバックカメラなどのカメラ機能と，ジャイロや加速度センサによるモーション機能を統合し，アプリケーションやゲームの中で拡張現実体験を生み出すことができる[4].

ARKitでは,TrueDepthカメラを使うことで顔の表情を検出し,さらにその表情を3DCGに対応させることで，自身の顔を拡張した仮想キャラクターに置き換える事ができる．本研究ではこの機能を用いたプログラムの作成及び動作に関する研究を行う.以下2章ではARとARKit,従来技術の説明及び比較について述べる．3章ではARKitを用いたFaceTrackingの方法及び，3DCGへの投影について説明する．4章では開発したプログラムを動作させ，表情との比較や動作を確認し，5章で結論を述べる．

2 章 AR と ARKit および従来技術

最近のスマホやタブレットのアプリには,普通のカメラアプリやゲームアプリだけでなく,カメラで撮影した映像とコンピュータが生成する仮想空間を高度に融合したアプリがある．これらのアプリの特徴は次のようなものである.

a. 現実空間にバーチャルなキャラクタが存在しているように感じさせる.
b. 自分自身が仮想世界のキャラクタになったような没入感を与える.
c. カメラで撮影した自分自身に装飾を行うなうことで仮想空間に入り込む.
d. 現実に仮想を持ち込んで拡張させる.

これらのことを実現する技術を Virtual Reality(VR, 仮想現実), Augmented Reality(AR, 拡張現実), Mixed Reality(MR, 複合現実), Substitutional Reality(SR, 代替現実)と呼ぶ．またはこれらは，総称して X Reality(xR, クロスリアリティ)と呼ばれている.

2.1 本研究で扱う AR

AR はコンピュータ技術による現実の拡張という意味であるが，研究や実用化が進んでいるものは，実写映像に CG 画像を違和感なく合成する技術である．この AR を大きく分けると,端末の GPS などから得るグローバルな位置情報に紐付けて情報を表示するロケーションベース型と，カメラで撮影した画像を使って，物体を認識したり局所的な空間情報を取得するビジョンベース型がある.本研究はビジョンベース型の中のマーカーレス型と呼ばれる手法にあたる.

マーカーレス型では，実空間に存在する机や椅子などの被写体を撮影した画像から，机の角のようなコーナー点や特徴点を抽出して画像認識や空間認識を行う．そして，その物体や空間に合わせた情報を付加して表示する.マーカーレス型はリアルタイムに実空間の解析を行い，特徴点を安定して認識・トラッキングする必要がある．従来は，これを行う計算コストの高さと処理の信頼性の低さが実用化の障害になっていた.しかし,コンピュータの処理能力が向上し，また，コンピュータビジョンのアルゴリズムが高度化したことで実用化が進んだ．また，TrueDepth カメラのような深度センサをはじめとする関連技術も開発が進み，ビジョンベース型かつマーカーレス型の AR がスマホやタブレット端末で利用可能になっている.

マーカーレス型のメリットは，特別なマーカーを用意することなく，現実の建物や風景に関連した付加情報を表示できることである.デザインやスペースが原因でマーカーの設置が難しい場合にも AR を用いた提示が可能である．デメリットとして，計算量が多く，処理の安定性と精度を確保することが難しい．従って，空間認識や物体認識に関

する専門的な知識やノウハウが必要になる事が難点である．しかし Apple 社の提供する ARKit を用いる事で，専門的な知識が必要な処理をこのフレームワークが行うため，比較的容易に AR アプリを実装することが可能になった．

2.2 ARKit

ARKit は iOS で動作するフレームワークの一つである．ここでのフレームワークはアプリケーションプログラムの開発に必要な API やクラスライブラリを関連する機能ごとにまとめたもののことである．2017 年に開催された世界開発者会議 WWDC2017 で，iOS11 の新機能として発表された．その後，2018 年に ARKit2，2019 年に ARKit3，2020 年に ARKit4 とバージョンアップされている．本研究では，ARKit2 以降に実装されている，目鼻口などの顔パーツを追跡する機能である Face Tracking(フェイストラッキング)を利用する．

2.3 Face Tracking

Face Tracking は，iPhone X あるいは iPad Pro（CPU が A12A）以降に備え付けられている TrueDepth カメラで顔を撮影することで顔の深度マップを測り，その形状を処理する事で，目鼻口などの顔部位の動きをトラッキングする技術である．これを用いる事で，顔の表情に合わせた 3DCG アニメーションや仮想的なフェイスペイントなどをリアルタイムで表示させることができる．本研究では，Face Tracking を使い，自作の 3DCG キャラクタに自身の表情の動きを対応させ，3DCG キャラクタの表情操作を行う．詳しい方法については３章で述べる．

2.4 従来技術

本研究の従来技術を説明し，本研究との違いを述べる．

- アニ文字

  iOS11 の新機能の一つとして登場した．容姿が決まったキャラクタが用意されており，それらのキャラクタに FaceTracking で得たデータで自分自身の顔の表情を投影することで，動きのある絵文字として表示する技術である．

図 3 iPhone の機能「アニ文字」の選択画面

- ミー文字

iOS12 で，アニ文字よりも自由度を拡張させた機能として登場した．ベースのキャラクタは用意されているが，髪型や帽子，眼鏡など顔のパーツ，装飾品などの特徴を変更することができる．自分仕様にカスタマイズしたアニ文字である．

図 4 ミー文字のアバタ作成画面の一部

- VRChat

  2017 年に，多人数参加型のソーシャル VR プラットフォームとして，VRChatInc からリリースされているプログラムである．25,000 を超えるコミュニティが作成されており，ワールドと呼ばれる仮想世界でのチャットルームに参加し，世界中の人々と交流が可能である．また Unity を利用することで，自身で用意したアバタ(3DCG のキャラクタ)で参加することも可能である.使用する機器によるが，リップシンクやアイトラッキング，まばたきも再現できる．

図 5 VRChat によるデモ動画のスクリーンショット

この他にも,2年前ほどから現れたバーチャルユーチューバーが有名になりつつある．これは 3DCG キャラクタを使い，そのキャラクタにロールプレイさせながら自分の作成した映像を配信するものである．バーチャルユーチューバーは 3DCG キャラクタと人間が一体化した AR 的な存在である．バーチャルユーチューバーの中には企業のタレントとして活動している人物もおり,そのような人物は企業が独自に開発したプログラムを利用して，ユニークなアバタになっている[9]．

本研究とこれらの従来技術の主な違いは,完全に自作の 3DCG キャラクタであるから,自分が好むどのような容姿にもなる事ができる．また，表情変化の程度を自由に設定できるので，自分が好む表情を 3DCG に投影することができる点にあり，これらの動作をこのプログラム１つで行えることにある．

3 章 ARKit を用いた FaceTracking と 3DCG への投影

ARKit の FaceTracking を利用して，3DCG に表情を投影する手順やデータ構造について説明する．なお，本研究は以下の開発環境で実施した．

開発環境：

MacBook Air 13-inch Early 2015

MacOS Catalina version 10. 15. 6

Xcode version 11. 4. 1

プログラミング言語：

Swift

使用端末(iOS デバイス)：

iPad Pro 11-inch version 13. 4. 1

3.1 FaceTracking の実装

顔を撮影しながらリアルタイムで顔の表情を追跡するための一般的な手順は，顔の検出，顔部位の識別，顔部位の追跡のように，複数のステップからなる．ARKit を用いると，これら処理の全てを図 6 に示す数行のコードで実装することができる．

```
59          //FaceTracking の開始
60          func resetTracking(){
61              //ARFaceTracking をサポートしているか
62              guard ARFaceTrackingConfiguration.isSupported else { return }
63              //FaceTracking を行うための設定
64              let configuration = ARFaceTrackingConfiguration()
65              //オブジェクトにシーンのライティングを渡すかどうか
66              configuration.  isLightEstimationEnabled = true
67              //セッションの開始
68              sceneView.session.run(configuration, options: [.resetTracking, .removeExistingAnchors])
69          }
```

図 6　顔追跡の設定を行うコードセグメント

図 6 において，62 行目は FaceTracking が利用可能なデバイスであることの確認である．ARKit の機能は AR セッションが実行されている時に利用できる．AR セッションには様々なコンフィグレーションがあり，その中の一つに FaceTracking が利用できるものがある．それは ARFaceTrackingConfiguration()関数を呼び出すことで利用可能になる（64 行目）．その後，68 行目で AR セッションを開始すると顔の検出が始まる．顔が検出されると，カメラに一番近く，大きく映る顔に対して，顔の位置・向き・表情などの情報を持つ ARFaceAnchor オブジェクトが生成される．

ARFaceAnchor オブジェクトのプロパティには，顔の姿勢・形状に関する geometry，表情変化を表現する blendShapes，目の動きを関する leftEyeTransform・

rightEyeTransform・lookAtPoint がある．この研究では blendShapes を使う．blendShapes は名前付き係数辞書（a dictionary of named coefficients）で，顔の特徴点の動きによって表情を表現している.名前付き係数は,左目に関して eyeBlinkLeft, eyeLookDownLeft など 7 個, 右目に関して eyeBlinkLeft, eyeLookDownRight など 7 個, 以下，口・あご・眉・ほお・鼻・舌に関したものがあり，合計 52 個ある．FaceTracking した結果，これらの名前付き係数に 0 から 1 の値がセットされる．

3.2 3DCG キャラクタの制作

自分を投影するアバタとなる 3DCG キャラクタを制作した．ここでは，3DCG 制作の流れを簡潔に述べる．3DCG の製作には Blender を用いた．

3DCG の制作手順は次の 3 段階からなる．

① メッシュの追加.

② メッシュの頂点, 辺, 面のいずれかを選択し, 伸ばしたり, 繋げて形を変形する.

③ ①と②の繰り返しによって，髪と顔をそれぞれ作成し，統合する.

これらの工程を説明する前に,「ポリゴン」と「メッシュ」を説明する.

- 「ポリゴン」は 3 点以上の頂点を結んだ多角形.
- 「メッシュ」は，三角形や四角形などのポリゴンの集合で面や立体の形状を定義したもの.

例えば，図 7 の立方体は，6 つの四角形ポリゴンを集めてできたメッシュで定義されている.

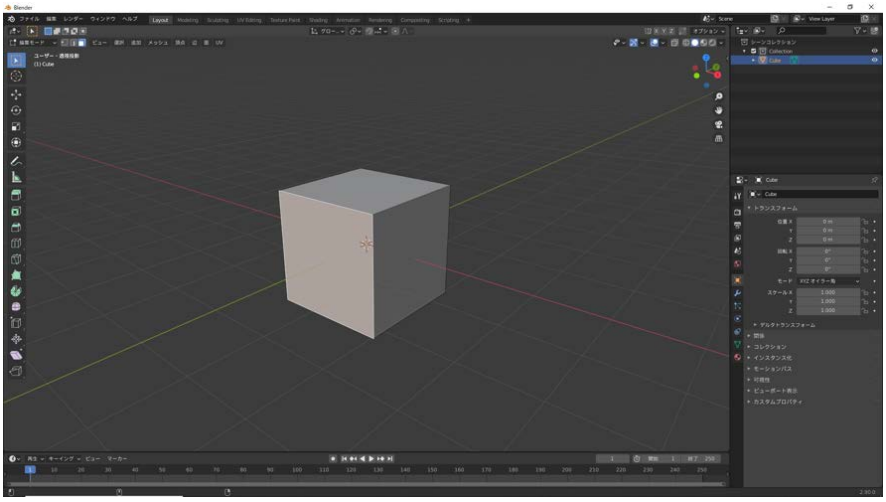

図 7 立方体を表現するメッシュ

3DCG の制作は様々なメッシュを組み合わせ，自身が求める形に近づけていく作業になる．滑らかな曲面を表現するには多数のポリゴンが必要になる．ポリゴン数が少なければ粗いモデルになる．ポリゴンが多いほど，情報量が増えるため，利用するコンピュータによっては動作が重くなる．今回の研究で作成した 3DCG モデルを顔と髪のパーツに分けて図 8 に示す．

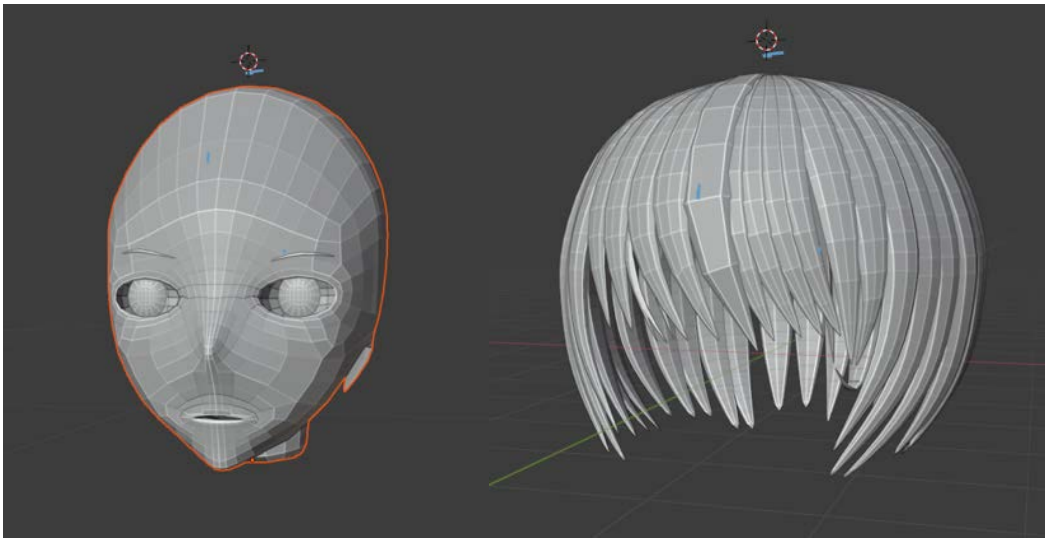

図 8　顔のメッシュ(左)と髪のメッシュ(右)

製作した顔の 3DCG モデルに表情変化を与えるために，Blender のシェイプキー機能を使う．シェイプキーは，3DCG のメッシュに変形のルールを対応させる機能である．シェイプキーを与えたメッシュは，0.0 から 1.0 の値に応じて，頂点の位置を制御できる．例えば図 9 は，まぶたのメッシュに対して，その制御量を 0，0.5，1.0 に設定した場合のメッシュ形状である．

(左)0．0　　(真ん中)0．5　　(右)1．0

図 9　シェイプキーの制御値とまぶたメッシュ形状の関係

FaceTracking で 3DCG を制御する場合，最大で 52 個のシェープキーを制御できる．今回は，その中で，目と口について，合計 24 個シェープキーを作成して，変形可能なように準備した．

3.3 表情の 3DCG への投影

図 10 のコードセグメントの，33 行目のようにして，シェイプキーを付与した 3DCG モデルをプログラムに入力する．入力ファイルの形式は Collada 形式を用いた．SCNScene()の引数として 3D モデルのファイルを渡すことで，これをシーンとして表示できるようになる．ここで，Blender の Collada 形式へのエクスポートが Xcode と相性が悪く，出力されるファイルの xml の記述に数カ所問題がある．これを解決するために，Jon Allee 氏がこれらの記述を訂正するプログラムを制作している．今回はこのプログラムを利用し，この問題を解決した[8]．

```
31          //シーンを作成して，SCNScene に．scn を設定
32          //""の中に．scn ファイルのパスを書く事でその．scn を取り込める
33          let scene = SCNScene(named: "art．scnassets/model．scn")!
```

図 10　3DCG モデルを取り込むためのコードセグメント

次に，シェイプキーと blendShapes の紐付けを図 11 のように実装した．TrueDepth カメラで FaceTracking した結果の ARFaceAnchor オブジェクト(101 行目 faceAnchor)から blendShapes を取り出し，key（顔の特徴点）と value（0 から 1 の値）ごとに，その値を 3DCG モデルの対応する名前の morpher にセットする．morpher は，Blender のシェープキーに対応し，0 から 1 の制御量でメッシュの形を変形できる．

```
 99          DispatchQueue．main．async {
100              //blendshapes の値を取得
101              let blendShapes = faceAnchor．blendShapes
102              //for 文を使って，各 blendsharps の値に対して，もし Float 型にダウンキャストが成功したら，
103              //指定された morpher のターゲット配列内のジオメトリに，指定された重み値を指定していく．
104              for ( key, value ) in blendShapes {
105                  if let fValue = value as? Float{
106              self．contentNode?．childNodes[0]．morpher?．setWeight(CGFloat(fValue), forTargetNamed: key.
rawValue)
107                  }
108              }
109          }
```

図 11　blendShapes とシェイプキーの紐付け

4 章 動作確認

４章では，実際の表情とそれを投影した 3DCG の表情を比較し，FaceTracking を使った 3DCG の制御が正常に動作していることを確認する．開発したデモアプリを動作させた時の，顔表情と対応する 3DCG の３つの例を図 12，13，14 に示す．

図 12　開眼で正面を向いた顔とその 3DCG モデル

図 12 左は正面・無表情・開眼状態の顔である．図 12 右の 3DCG もそのようになっているので，これは正しい動作である．

図 13　視線を横に向けた顔とその 3DCG モデル

図 13 左は正面・無表情・視線を左に向けた顔である．図 13 右の 3DCG では，左目の視線は右を向いているが，右目は動きが少ないようである．3DCG でもある程度は視線が右に向いているが，人間の表情ほどには，視線の向きを感じることはできない．

図 14　目を閉じた顔とその 3DCG モデル

図 14 左は左目を閉じた表情である．図 14 右の 3DCG でも左目が完全に閉じているので，正しく動作していることがわかる．

このように，FaceTracking の機能と，それを用いた 3DCG の表情コントロールが概ね正しくどうしていることがわかった．ここで，図 13 において，3DCG の右目の動きが少ないように見えるのでそのことを検証する．図 15 の左は図 13 の右と同じものである．図 15 の右は，Blender 内で右目・左目のシェープキーを，最大限，左側を向くように調整した 3DCG モデルである．この両者を見比べると，図 15 の左の表情もシェイプキーの最大値に近い表情であることから，FaceTracking とその 3DCG への投影は，この場合でも正しいと考えられる．

図 15　図 13 のデモの 1 シーン(左)と Blender 内の表情(右)

ここで人の表情である，喜怒哀楽より，喜びと怒りの２つの表情を実際の顔と 3D モデルで比較した．

図 16　喜びの表情

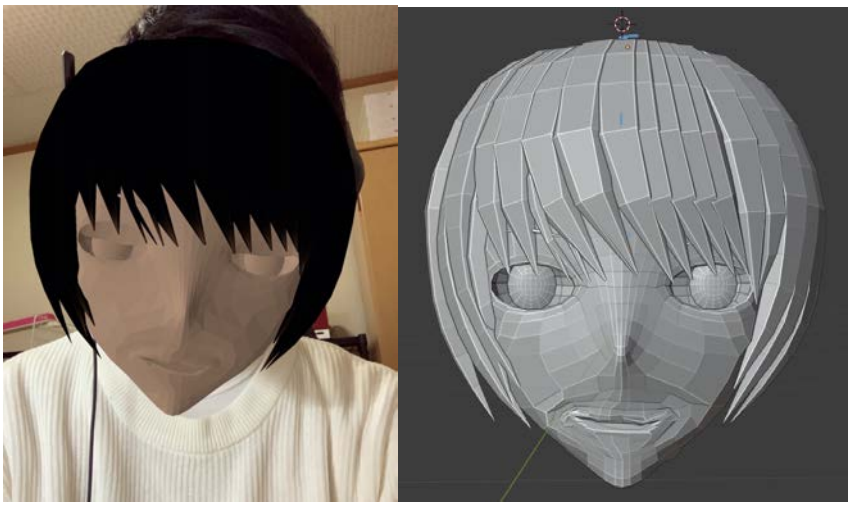

図 17　動作中の喜びの表情(左)と設定した喜びの表情(右)との比較

図 16 ではまぶたの部分が少し下に下がっているが，口の表情は本来設定していた表情(図 17)とほぼ同じように動作をしていた．口の動きは，ほぼ最大値の形をしている．次に怒りの表情を見ていく．

図 18　怒りの表情

図 19　動作中の怒りの表情(左)と設定した怒りの表情(右)との比較

図 18 では眉の下がり具合に合わせてまぶたが鼻筋に近いところから下がり，怒りを表した表情をしている．またこの表情も設定した表情と比較を行うと，顔の向きが反映されて見づらいものの，まぶたの下がりぐらいが同じことが見える．これらの事から，正常に動作している事，そして，3D モデルの表情操作が行えていることがわかる．

5 章 結論

本研究によって，iPad Pro に備え付けられている TrueDepth カメラを用いて FaceTracking を行い，自作した 3DCG モデルに表情を投影することができた．このプログラム開発を通じて，ARKit を利用することで，高度な専門知識が必要な FaceTracking による 3D モデルの制御が比較的容易に実現可能であることがわかった．FaceTracking は，基本的に，これを実行する AR セッションを開始するだけで可能になり，その結果は，ARFaceAnchor オブジェクトのプロパティに自動的に保存されていた．その中で顔の表情は blendShapes という名前付き係数辞書に，52 個の顔特徴点の名前とその変化量として保存されていた．また本研究を実施したことで，FaceTracking 技術以外にも，Swift 言語，3DCG 制作ソフト Blender を扱えるようになった．

本研究で不十分なことの一つは，他の類似技術と比較した時に，明確にオリジナリティがあると断言できる箇所が少ないことである．オリジナリティのあるものを制作するのであれば，AR 技術や Swift 言語，iOS の様々なフレームワークについて深く理解する必要があると考える．しかし，すでに何かオリジナルなアイデアがあり，それを AR アプリとして開発するのであれば，ARKit などのフレームワークを活用することで，数ヶ月程度の短期間で，実用的なアプリの開発が可能である．

参考文献

[1] 鴻池賢三, 「「AR=拡張現実」とは何か？VR との違いは？実現するちょっと未来のカタチ」, 価格．com マガジン, 2017．1．18,
URL = https://kakakumag．com/pc-smartphone/?id=9609 (2021．1．12 現在)

[2] div,Inc, 「TECH CAMP,【誰でも分かる】AR(拡張現実)の仕組み，技術」,2017．10．4,
URL = https://tech-camp．in/note/technology/18791/#AR2 (2021．1．12 現在)

[3] Apple Inc,「先進の Face ID テクノロジーについて」,
URL = https://support．apple．com/ja-jp/HT208108 (2021．1．12 現在)

[4] Apple Inc, 「ARKit – 日本語ドキュメント AppleDocument」,
URL = https://developer．apple．com/jp/documentation/arkit/ (2021．1．12 現在)

[5] VRChat Inc,「VR Chat」,URL = https://hello．vrchat．com (2021．1．12 現在)

[6] Blender の易しい使い方, URL = https://blender-cg．net/polygon-mesh-surface/#toc3 (2021．1．13 現在)

[7] YUNODA, 「Blender のシェイプキーを完全に理解する」, 3DCG school, 2020.5.18,
URL = https://3dcg-school．pro/blender-shape-key/#i (2021．1．13 現在)

[8] Stack Exchange Inc,「How do I load SCNMorpher targets from a Collada ．dae file's morph controllers into SceneKit?」, 2016．5．30,
URL = https://stackoverflow．com/questions/37516117/how-do-i-load-scnmorpher-targets-from-a-collada-dae-files-morph-controllers-in (2021．1．13 現在)

[9] Rentio Inc. , 「いまさら聞けない「バーチャルユーチューバー(Vtuber)」とは？VR の新しい体験をご紹介」, 2020．3．4, URL = https://www.rentio.jp/matome/2019/07/vtuber-beginner/ (2020．1．29 現在)

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします．また助言を頂いた学生の方々に感謝いたします.

付録

- AppDelegete．swift
  - 本研究ではストーリーボードを利用して制作を行ったため，特に説明を行う必要がある機能はない．
- ViewController．swift
  - 本研究の制作したアプリの根幹を成すファイルであり，説明が必要な箇所の動きを下記で述べる．

・ resetTracking

―この関数では，セッションを開始するため，まず FaceTracking が行えるかを確認し，可能であればセッションに FaceTracking を行うための準備を行わせ，準備が完了できればセッションを開始させる．

・renderer

―この名前の関数は２つ存在し，１つ目で TrueDepth カメラの深度マップから作成された ARAnchor を取得し，Content Node を通して root Node に戻り値として返し，スクリーンに表示を行うために 3DCG モデルの向きや表情のデータの準備を行う．2 つ目は，ARAnchor を取得し，Blendshape と紐付けさせることで表情の投影を行う．

- DepthValueCheck．swift
  - 毎フレームごとにデプスデータを得られることができるファイルである．本研級では利用していないため，説明はない．